FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

A Scalable, Self-organizing Communications System for very large Wireless Sensor Networks

Mohammad Mahmoud Abdellatif

Programa Doutoral em Telecomunicações

Orientador: Doutor José Manuel Soares Oliveira, da Faculdade de Economia da Universidade do Porto

Co-Orientador: Doutor Manuel Alberto Pereira Ricardo, do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

9 de Março de 2015

A Scalable, Self-organizing Communications System for very large Wireless Sensor Networks

Mohammad Mahmoud Abdellatif

Programa Doutoral em Telecomunicações

Aprovado em provas públicas pelo Júri:

Presidente: Doutor José Alfredo Ribeiro da Silva Matos, Professor Catedrático da Faculdade de Engenharia da Universidade do Porto

Arguente: Doutor Jorge Sá Silva, Professor Auxiliar do Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Arguente: Doutor Joel José Puga Coelho Rodrigues, Professor Auxiliar do Departamento de Informática da Universidade da Beira Interior

Vogal: Doutor Adriano Jorge Cardoso Moreira, Professor Associado do Departamento de Sistemas de Informação da Escola de Engenharia da Universidade do Minho

Vogal: Doutor Vítor Manuel Grade Tavares, Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Vogal: Doutor José Manuel Soares Oliveira, Professor Auxiliar da Faculdade de Economia da Universidade do Porto (Orientador)

18th of September 2015

The best way out is always through.

Robert Frost.

Acknowledgments

First and foremost I would like to thank Allah for His kindest blessings on me and all the members of my family. And for always helping me find the right path even when I felt that there is none.

My deep appreciation goes to my advisor Prof. José Manuel Oliveira. He was always there when I needed him ever since the day I arrived to Porto. I am extremely grateful to him for his prompt replies to my questions and his numerous proofreads of all the work that was done during the Phd. I am thankful to his help and support inside and outside the work.

I would like also to thank my other advisor Prof. Manuel Ricardo. I appreciate all his contributions of time, ideas, even with his busy schedule, he could always find the time to discuss the work and give me his feedback which has always resulted in the improvement of the work.

The help and support of my research institute INESC TEC can not go without notice. It has been a privilege to work in this environment that is dedicated to research and to help young researchers achieve their potential. I had many discussions with my colleagues here that helped me improve my work and I am really thankful to all their help and friendship. I like to thank: Bruno Marques, Dossa Massa, Helder Fontes, Jaime Dias, Muhammad Ajmal, Nuno Salta ("RIP"), Pedro Fortuna, Pedro Pinto, Pedro Silva, Renata Rodrigues, Rui Campos, Saravanan Kandasamy. Apologies to anyone I forgot.

Thanks to Faculdade de Engenharia da Universidade do Porto for providing excellent education and accepting me as student. It is truly one of the best faculties that I had the privilege to study in. This work is funded by the ERDF through the Program COMPETE and by the Portuguese Government through FCT-Fundação para a Ciência e Tecnologia, project ref. CMU- PT / SIA / 0005 / 2009 and by the research grant number SFRH / BD /

68759 / 2010.

I can not forget to thank my friend and neighbor Hazem Ali for his support and friendship during my years here in Portugal. Many things would have gone badly in my life if it weren't for his help. I would also like to thank Rita Pacheco for her help with the abstract translation.

Last but not least, I humbly offer my sincere thanks to my parents for their incessant inspiration, blessings and prayers. They never gave up on me and they always pushed me forward. I don't think I would have been able to finish if it weren't for their support and for that I am eternally grateful.

Mohammad Mahmoud Abdellatif

Resumo

As redes de sensores sem fios (Wireless Sensor Networks, WSNs) são formadas por uma grande quantidade de nós de dimensões reduzidas que são capazes de detetar mudanças num ambiente e comunicá-las através da rede. Um exemplo deste tipo de rede é uma central fotovoltaica (FV), onde existe um sensor ligado a cada painel solar. Cada sensor é responsável por detetar a produção do painel, que é depois enviada para um nó central para ser processada.

Apesar das redes de sensores sem fios transmitirem dados em baixo débito, estas apresentam ainda outros desafios que estão maioritariamente ligados à eficiência das comunicações e à eficiência energética. Além disso, à medida que a rede cresce, torna-se impraticável ou mesmo impossível configurar todos estes nós manualmente. Portanto, neste contexto é essencial usar algoritmos de auto-organização e configuração. Mais ainda, devido à caraterística "muitos para um" (many-to-one) da comunicação nas redes de sensores sem fios, assim como as interferências ecolisões, a recolha programada de dados torna-se um grande desafio que deve ser cuidadosamente analisado.

Este trabalho aborda dois problemas: a autoconfiguração de nós e a recolha de dados a partir desses nós. Para o problema da autoconfiguração de nós são propostos três algoritmos que permitem aos nós da rede identificar automaticamente os seus vizinhos mais próximos, a sua posição relativa na rede, e selecionar o canal de frequência em que devem operar. Para isso, é utilizado o valor da indicação do nível de sinal recebido (Received Signal Strength Indicator, RSSI) das mensagens enviadas e recebidas durante a fase de configuração. O desempenho destes algoritmos é testado através de simulações e testbeds. Foi possível concluir que os algoritmos permitem que os nós se autoconfigurem sem qualquer interferência do operador da rede. Os resultados demonstram que o erro ao nível do desempenho decresce à medida que se aumenta o número dos valores de RSSI

utilizados para a tomada de decisões. Além disso, o número de nós na rede afeta consideravelmente o erro de configuração. No entanto, o valor do erro é ainda aceitável mesmo para um grande número de nós simulados.

Relativamente ao problema da recolha de dados, foi avaliado o desempenho de três técnicas diferentes de recolha, com e sem agregação de dados. Este estudo considerou uma rede sem fios com um número fixo de nós, utilizando diferentes valores de tráfego oferecido, tendo sido possível estimar a capacidade de transmissão da rede para cada técnica, assim como para cada valor de tráfego oferecido.

Os resultados demonstram que à medida que o tamanho da rede aumenta, a técnica selecionada tem um melhor desempenho em termos de perda de pacotes e capacidade de transmissão. No entanto, isto irá levar a um maior atraso ao nível do processo de recolha de dados. Além disso, com a agregação de dados numa rede de múltiplos saltos (multi-hop), a técnica selecionada tem um desempenho semelhante ao das outras técnicas sem agregação numa rede de um só salto (one-hop). Desta forma, decidiu-se usar uma rede de múltiplos saltos por forma a reduzir as gamas de transmissão e os níveis de interferência entre os diferentes nós. Com o desempenho da técnica selecionada a ser semelhante ao das restantes em termos de capacidade de transmissão numa rede de múltiplos saltos, foi possível demonstrar que esta é a melhor escolha para os requisitos da rede.

Abstract

Wireless Sensor Networks (WSNs) are made of a large amount of small devices that are able to sense changes in the environment and communicate these changes throughout the network. An example of such network is a photo voltaic (PV) power plant, where there is a sensor connected to each solar panel. The task of each sensor is to sense the output of the panel which is then sent to a central node for processing. Even though low data rate is employed in WSNs, other challenging issues appear in the network.

Challenges are mostly related to the reliability of the communication links and to the energy efficiency. Moreover, as the network grows, it becomes impractical and even impossible to configure all these nodes manually. The use of self-organization and auto-configuration algorithms becomes essential in this context. Also, because of the "many-to-one" feature of the communication in WSNs, the wireless interferences and the collisions, the scheduling of data collection becomes a challenging problem which needs to be carefully addressed.

This work addresses these two problems: the self-configuration of nodes and the data collection from nodes. For the self-configuration problem, three algorithms are proposed that allow nodes in the network to automatically identify their closest neighbors, their relative location in the network, and select which frequency channel to operate in. This is done using the value of the Received Signal Strength Indicator (RSSI) of the messages sent and received during the setup phase. The performance of these algorithms is tested by means of both simulations and testbed experiments. We were able to conclude that the algorithms allow nodes to configure themselves with no interference from the operator of the network. Results showed that the error in performance decreases as we increase the number of RSSI values used for decision making. Additionally, the number of nodes in the network affects the setup error greatly. However, the value of the error is still

acceptable even for high number of simulated nodes.

As for the data collection problem, we have evaluated the performance of three different data collecting techniques with and without data aggregation. The study considered a wireless network with fixed number of nodes using different values of the offered load, estimating the network throughput for each technique and offered load.

Results showed that as the size of the network grows, the selected technique performs best in terms of packet loss and throughput. However, this will lead to having a larger overall delay for the data collection process. Moreover, with data aggregation in a multi-hop network, the selected technique performs close to the other techniques with no aggregation in a one-hop network. We have decided to use a multi-hop network in order to reduce transmission ranges as well as interference levels among the different nodes. And with selected technique performing close to the others in terms of throughput in a multi-hop network we were able to show that it is the best choice for the requirements of the network.

Contents

List of Figures

List of Tables

Abbreviations

6LoWPAN	IPv6 over Low Power Wireless Personal Network
AMA	Advanced Metering Infrastructure
AWGN	Additive White Gaussian Noise
CoAP	Constrained Application Protocol
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CRC	Cyclic Redundancy Check
DAG	Directed Acyclic Graph
DAO	Destination Advertisement Object
DIO	DODAG Information Object
DODAG	Destination-Oriented Directed Acyclic Graph
DSSS	Direct Sequence Spread Spectrum
ETX	Expected Transmission Count
FFD	Full Function Device
FHSS	Frequency Hopping Spread Spectrum
GTS	Guaranteed Time Slot
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IFS	Inter-Frame Space
INESC	Instituto de Engenharia de Sistemas e Computadores
IP	Internet Protocol
IPHC	Internet Protocol Header Compression

LLN	Low Power, Lossy Network
LQI	Link Quality Indicator
LR-WPAN	Low Rate Wireless Personal Area Networks
MAC	Medium Access Control
MPDU	MAC Protocol Data Unit
MRM	Multi-path Ray-tracer Medium
MTU	Maximum Transfer Unit
OFDM	Orthogonal Frequency Division Multiplexing
OSI	Open Systems Interconnection
PAN	Personal Area Network
PCF	Point Coordination Function
PHY	Physical Layer
PV	Photo Voltaic
RFC	Request for Comments
RFD	Reduced Function Device
RFID	Radio Frequency Identification
RPL	Routing Protocol for Low Power Lossy Networks
RSSI	Received Signal Strength Indicator
SICS	Swedish Institute of Computer Science
SELF-PVP	Self-organizing power management for photo-voltaic power plants
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UDGM	Unit Disk Graph Medium
WPAN	Wireless Personal Area Networks
WSN	Wireless Sensor Network

Chapter 1

Introduction

Wireless Sensor Networks (WSNs) have been a hot research topic in the recent years. They consist of a large amount of small devices that are able to sense changes in the environment and communicate these changes throughout the network [1]. WSN applications are various, they are used in smart-cities, environmental monitoring, distributed sensing in industrial plants, and health care [2].

Even though low data rate is employed in WSNs, other challenging issues appear in the network. Challenges are mostly related to the reliability of the communication links and to the energy efficiency [3]. Also, because of the "many-to-one" feature of the communication in WSNs, the wireless interferences and the collisions, the scheduling of data collection becomes a challenging problem which needs to be carefully addressed.

1.1 Motivation

The need for an alternative and clean power source has grown continuously over the past years since the world is still mostly dependent on the use of fossil fuel which has a lot of negative effects on the environment. One of the main sources of alternative energy is solar power, which has proven to be a good choice to replace fossil fuel. However, even with the current technology, and with a large solar power plant, the efficiency of the power generation is not sufficient. An example of a photo-voltaic (PV) power plant is the

Almaraleja power plant, in Alentejo, Portugal, in which the number of solar panels could go above 200,000 spread over a large area (250 hectares).

The main motivation that drives this PhD research is to achieve an increase in power efficiency in photo-voltaic power plants. The increase will be achieved using a novel, distributed, real time adaptive network controller of sensors/actuators, where a sensor node is connected to each solar panel in the network, to bring optimality to the overall power output of the panels' array. The small and low power wireless sensor nodes will sense local variables, communicate among each other these variables and compute local errors with the goal of optimizing, in real time, the overall performance of the panels' array. Our work is focused on allowing those many sensors nodes to communicate with each others reliably in order to achieve this goal.

1.2 Problem Statement

The first problem that arises is that as this communications network grows, it becomes impractical or even impossible to configure all the nodes in the network manually. In order to solve this problem, the use of self-organization and auto-configuration algorithms becomes essential.

Secondly, the interference among the nodes and the large set of neighboring nodes will increase as the number of nodes continue to increase. Interference will increase the rate of frame retransmissions. and to reduce the reliability of the network.

Furthermore, sensor data has to be collected from nodes in a way that is reliable regardless of the topology of the network which can be very dense with a very high number of nodes.

1.3 Objectives

The main objective of the PhD is to develop a complete, scalable, self-organizing communications solution that can be used to interconnect a large and dense network of sensors.

The goal is to add self-organization and scalability to a distributed communication network, with the ability to sense and control the operation of a dense wireless sensor network.

Due to the specificity of the sensor network (heterogeneous data, real time constraints, several communication modes - broadcast and unicast, grid topology) a dedicated wireless system design will have to be undertaken. Although the research here presented has a key application in mind - the power optimization in PV power plants - we envision many other applications - tackling generic global optimization with sensor networks.

This PhD focuses on solving the problem of the self-organization of sensors in clusters, which also includes their auto-configuration and frequency planning. Additionally, we tackle the efficient broadcast based data collection and communications in a large network. Using simple communications techniques for the transmission and routing of data, while preserving some degree of reliability.

1.4 Overview of the Proposed Network Topology

The proposed WSN topology is shown in Figure 1.1. Such setup is referred to as a strip-based deployment and it allows for full coverage of the area under consideration. In a real life scenario, the nodes within a column are placed side by side with a distance of about 2 meters between each consecutive sensor, while columns have a more wider distance between them in order to allow maintenance access.

In order to reduce the interference between columns, each column is configured to operate in a frequency channel that is different from the other columns. In order to enable inter-column communication, a core network is created, operating in a frequency different from all the columns. Core network nodes are placed between columns to enable inter-column communication, and when the need for inter-column communication arises the respective column heads will have to switch to this core frequency and communicate through the core network. A group of columns with their access network can be considered as a network building block which can be replicated as the network grows.

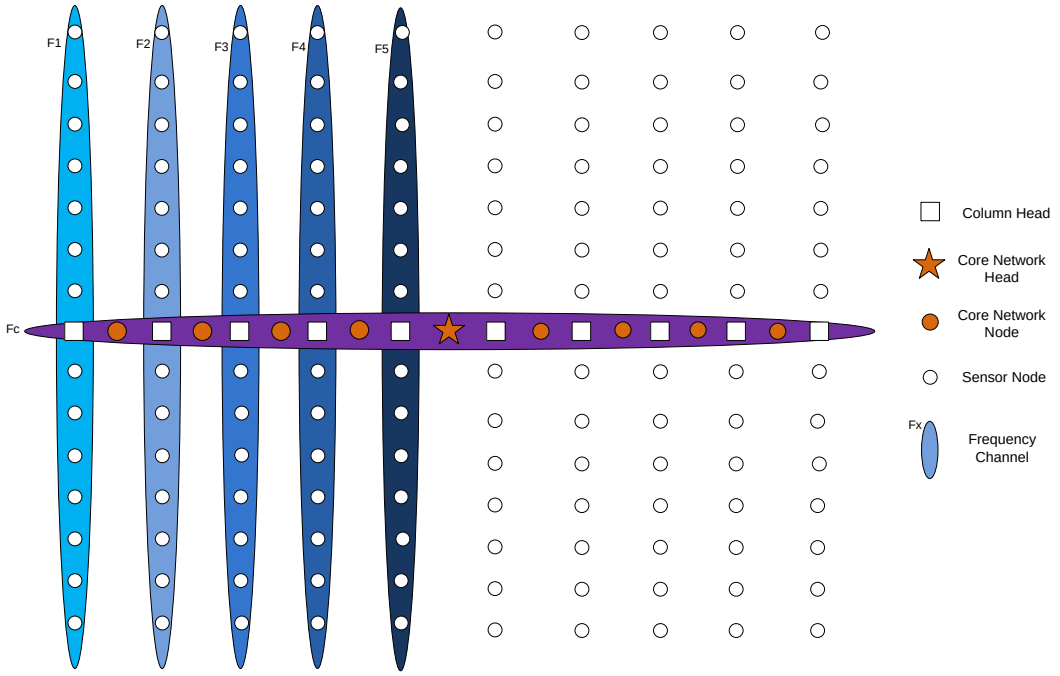


Figure 1.1: Network topology.

1.5 Main Contributions

This PhD thesis has two main contributions, which are described in the following subsections.

1.5.1 Self-Configuration of Nodes

Three novel algorithms were proposed that, when run consecutively, allow each node in the system to identify its neighbors, locate its relative location in the network, and select its operating frequency channel in order to reduce the overall interference among nodes.

These algorithms were designed for this specific network where nodes are placed in a static grid topology and where nodes communicate mainly with their neighbors within the same column. There are related algorithms proposed in the literature [4, 5]. However, none of them meet the needs of the objectives mentioned above.

- **Neighbor Identification Algorithm.** Allows each node in the network to find its closest neighbors, i.e. neighbors which can be reached directly without multi-hopping. The algorithm works by measuring the value of the Received Signal Strength Indicator (RSSI) of the ping messages that are sent back and forth between

nodes in the network after booting. Using this information, nodes calculate which neighbor(s) has the highest RSSI value. These neighbors will be considered then as the closest neighbors in the network. Nodes are required to know their closest neighbors in the network in order to limit the communication between these nodes and so, reduce any unwanted traffic and in turn leads to less interference, and less energy consumption, which will lead to the ability of adding more nodes to the network.

- **Relative Location Algorithm.** Used to enable each node to find its relative location in the network with respect to the other nodes and decide its role in the network based on that location. The role of each node will dictate how will it behave throughout its lifetime. The node's role is determined with no interference from the network operator with all the nodes having the same code and the same hardware.

- **Frequency Allocation Algorithm.** This algorithm is used to divide the network into small networks of columns where every column operate in a different frequency channel. The allocation is done in order to reduce the interference among the nodes as well as to enable the scalability of the solution. The core network will be still operating in the initial frequency in order to allow nodes in different columns to communicate with each others if needed.

These three novel algorithms allow the complete and automatic self-configuration of nodes in a static network with grid topology with no interference from the network operator. No other solution was found in literature that was able to achieve these goals for the specific network proposed in this work.

1.5.2 Aggregated Data Collection Technique

The second original contribution of this PhD thesis is a reliable data collection technique that employs aggregation, polling, and message scheduling to collect the data from nodes within their column. It is designed in order to reduce the number of collisions/retransmissions of messages between the nodes. The data collection technique starts running as soon as the network is divided into a group of smaller networks of columns.

Using the proposed polling technique to collect data from the nodes increases the end to end delay, however, as the rate of change in the state of the network is slow, this level of delay can be considered as acceptable. The proposed technique was compared with other existing techniques in literature and was proven, in the specific network we are testing, to further reduce the traffic being sent back and forth between the nodes, and increases the reliability of the network.

1.6 Overview of Published and Submitted Papers

- Published a conference paper entitled "**Impact of Data Collecting Techniques on the Performance of a Wireless Sensor Network**", in Proceedings of the ISWCS 2012, the Ninth International Symposium on Wireless Communication Systems, Paris, France, August 28-31 [6].

- Published a conference paper entitled "**The Effect of Data Aggregation on the Performance of a Wireless Sensor Network Employing a Polling Based Data Collecting Technique**", to the Wireless Days 2013 conference WD'13, Valência, Spain, November 13-15 [7].

- Published a conference paper entitled "**Neighbors and Relative Location Identification Using RSSI in a Dense Wireless Sensor Network**", to the 13th Annual Mediterranean Ad Hoc Networking Workshop, Piran, Slovenia, June 2-4, 2014 [8].

- Submitted a journal paper entitled "**The Self-Configuration of Nodes Using RSSI in a Dense Wireless Sensor Network** to the Telecommunication Systems journal. Submitted on the 15th of October 2014. Currently under review.

1.7 Structure of the Thesis

The rest of the thesis is organized as follows. Chapter 2 gives a brief background on the technologies used in the scope of this thesis and discusses some of the related work

already published in literature. Chapter 3 describes in detail the work done for the self-configuration of nodes in dense wireless sensor networks. Chapter 4 focuses on the work done on the data collection techniques. Finally, thesis conclusions and ideas for future work are listed in Chapter 5.

Chapter 2

Background and Related Work

In this chapter, we give a brief summary of the technologies that we are using in the scope of this thesis and we list some of the related work that could be found in the literature.

2.1 Background

In this section, we discuss in details some of the protocols and technologies that we used to achieve the goals of this thesis. Some of these protocols include the Internet protocol suite and the different stack protocols used. We also discuss the operating system, the simulation environment as well as the hardware platform that we used to carry on the analysis and simulation of the proposed algorithms.

2.1.1 The Internet Protocol Suite

The Internet Protocol Suite is a set of communication protocols grouped for the first time in RFC 1122 [9] and RFC 1123 [10]. It is also known as TCP/IP protocol stack due to the division into abstraction layers of the communications suite. The information flows in both directions, but each layer communicates only with the layer immediately above or beneath. This is done by encapsulating the data on the way down and de-encapsulating it on the way up. In order for this communication to happen, each layer requires the layer underneath to meet certain requirements. At the same time, each layer should fulfill the requirements of the layer immediately above.

The Internet Protocol Suite is divided into four abstraction layers according to RFC 1122: Link Layer, Internet Layer, Transport Layer, and Application Layer. However, due to the usual mapping of the TCP/IP stack onto the International Standards Organization's Open System Interconnect (OSI) model, it is also common to refer to the Physical Layer as a hardware layer at the lowest part of the Link Layer. Figure 2.1, illustrates the TCP/IP protocol stack, including the Physical Layer within its lowest level.

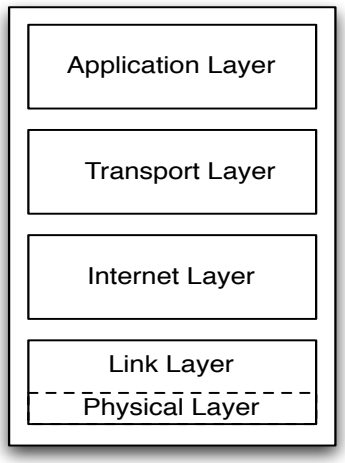


Figure 2.1: The TCP/IP stack.

Link Layer Groups the different protocols that operate only between adjacent nodes in the same link segment.

Internet Layer Is the set of protocols in charge of delivering packets from the originating host to the destination, traversing different networks if necessary. Due to the mapping onto the OSI model, it is also commonly referred to as "Network Layer".

Transport Layer Provides convenient services such as connection-oriented data-stream support, reliable end-to-end communication, flow and congestion control, and host-level multiplexing.

Application Layer The set of protocols involved in the process-to-process communication.

2.1.2 IEEE 802.3

IEEE 802.3 [11] is a IEEE working group and a set of standards, and not only a single standard as is usually thought of. There are several versions and amendments that were added to it, with IEEE 802.3-2008 being the latest revision. IEEE 802.3-2008 defines the physical layer and the data link layer's media access control (MAC) of a wired Ethernet. As for the physical layer, this family of standards supports several types of media, such as different types of coaxial cable, shielded and unshielded twisted pair, and fiber-optic cables. The supported transmission data rates range from 10 Mbit/s to 100 Gbit/s. Some media support half or full-duplex transmission. The MAC protocol specified in IEEE standard 802.3 is Carrier Sense Multiple Access with Collision Detection (CSMA/CD). This MAC protocol was utilized in the experimental Ethernet developed at Xerox Palo Alto Research Center. However, new implementations operating in full-duplex mode no longer utilize CSMA/CD since in full-duplex mode for a point-to-point link there is no probability of collisions. This MAC layer consists of the channel-access portion of the link layer used by Ethernet, but it does not define a logical link control protocol (general implementations use the IEEE 802.2 logical link layer). Additionally, the standard defines the mapping between IEEE 802.3 MAC service interface primitives. As result, Ethernet's data link-layer protocol can be encapsulated within the MAC Client Data field of IEEE 802.3 packets (the common set of service interface primitives enables bridging between IEEE 802 MAC/PHY protocols). Figure 2.2 shows this Ethernet data link-layer into IEEE 802.3 MAC Client Data field encapsulation.

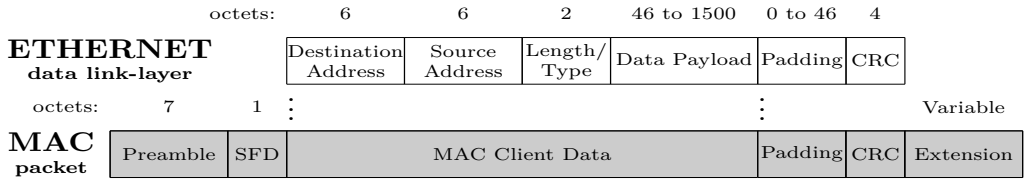


Figure 2.2: Ethernet data link layer protocol encapsulated into the MAC Client Data field of a IEEE 802.3 MAC packet.

Preamble Used for synchronization between the sender and the receiver.

SFD Start of Frame Delimiter. Indicates the end of the preamble and the start of the packet data. It has a constant value of 0xAB.

Destination Address 48-bit IEEE 802.3 MAC address of the destination of the frame.

Source Address 48-bit IEEE 802.3 MAC address of the originator of the frame.

Length/Type This field can have two different meanings. If its value is greater than 1500, then it indicates the type of the upper-layer packet being transported. If the field value is less than or equal to 1500, it indicates the length of the payload.

Data Payload The data being transmitted.

Padding Optional Padding. This field is required if the total Ethernet frame length is less than 64 bytes.

CRC Cyclic Redundancy Check for integrity verification.

Extension Optional field included only in half-duplex operation when the frame is shorter than the CSMA/CD slot time.

2.1.3 IEEE 802.15.4

The IEEE 802.15.4 standard [12] specifies the physical and media access control (MAC) for low-rate wireless personal area networks (LR-WPANs). Although LR-WPANs fall within the wireless personal area networks (WPANs) family of standards, they may extend the personal operating space. An LR-WPAN is a simple, low-cost wireless communication network optimized for use in applications with limited power and low throughput requirements.

LR-WPANs main aims are low power consumption and low cost, while maintaining a reliable data transfer, short-range communication link, and simple, flexible protocol.

2.1.3.1 Topologies

The IEEE 802.15.4 standard defines two different device types: full-function devices (FFDs) and reduced-function devices (RFDs). FFDs can participate in the Personal Area Network (PAN) as a PAN coordinator, a coordinator, or as a normal device. Even though a network may consist of just RFDs, the presence of at least one FFD acting as a PAN coordinator is recommended.

An LR-WPAN may operate in either peer-to-peer or star topologies. In a star topology, all the communication between devices must pass through the central node, which is the PAN coordinator. The PAN coordinator is responsible for initiating, routing, and terminating the communication in the network. On the other hand, in a peer-to-peer network, communication between any two nodes is possible as long they are in range; this topology offers greater flexibility, allowing all sorts of mesh formations, but at the cost of increased node power consumption. Peer-to-peer topologies require a PAN coordinator; however they are also likely to require a suitable routing protocol in case multi-hop is needed. This routing protocol should be provided by the upper layers and hence is beyond the scope of the IEEE 802.15.4 standard.

2.1.3.2 Physical Layer

Since its release in 2003, different amendments have been defined adding new possible physical layers and/or extending the capabilities of the previously defined ones. The main different unlicensed frequency bands and modulations, together with the supported data rates defined by the IEEE 802.15.4 physical layer are shown in Table 2.1.

Table 2.1: IEEE 802.15.4 physical layers

Physical Layer (MHz)	Frequency Band (MHz)	Modulation	Bit rate (kb/s)
868/915	868 - 868.6	BPSK	20
	902 - 928		40
868/915	868 - 868.6	ASK	250
	902 - 928		250
868/915	868 - 868.6	O-QPSK	100
	902 - 928		250
2450	2400 - 2483.5	O-QPSK	250
UWB	250 - 750	BPM-BPSK	851,110,
	3244 - 4742		6810, and
	5944 - 10234		27240
2450 (CSS)	2400 - 2483.5	DQPSK	250, 1000
780	779 - 787	O-QPSK	250
780	779 - 787	MPSK	250
950	950 - 956	GFSK	100
950	950 - 956	BPSK	20

2.1.3.3 Medium Access Control (MAC)

The IEEE 802.15.4 MAC layer is responsible for the following tasks:

- Beacon management;

- PAN association and disassociation;
- Employing the CSMA-CA mechanism for channel access;
- Handling and maintaining the Guaranteed Time Slot (GTS) mechanism;
- Frame validation;
- Acknowledged frame delivery;
- Supporting device security.

In star-topologies, the IEEE 802.15.4 MAC layer provides a beacon-based synchronization mechanism for data transmission and reception between devices and the PAN coordinator, which permits nodes to only listen to the channel at regular intervals, allowing for power saving. In peer-to-peer topologies, however, this synchronization mechanism is not provided by the standard and, if required by specific applications, it needs to be implement at upper layers.

The MAC layer defines four different types of frames: beacon frames, acknowledgment frames, MAC command frames, and data frames. Beacon frames are used in the synchronization mechanism. Acknowledgment frames, whose use is optional, are used to acknowledge transmissions. MAC command frames carry protocol commands, such as "Association request", or "Data request". Finally, data frames are used for all transfers of data. Figure 2.3 illustrates the structure of a data frame. The fields of that frame are explained in more details next.

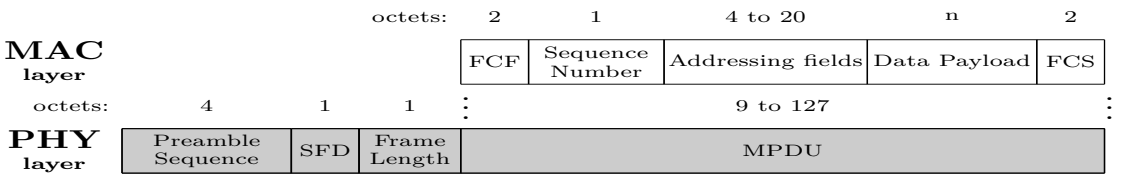


Figure 2.3: IEEE 802.15.4 data frame.

Preamble Sequence Used to obtain chip and symbol synchronization with an incoming message. It is composed of 32 binary zeros.

SFD	Start of Frame Delimiter. Indicates the end of the preamble and the start of the packet data. It has constant value of 0xE5.
Frame Length	Length of the MAC protocol data unit (MPDU).
FCF	Frame Control Field. Contains information defining the frame type, addressing modes, and other control flags.
Sequence Number	Used to match acknowledgment frames to data or MAC command frames.
Addressing Fields	The IEEE 802.15.4 standard supports short (16 bit) and long (64 bit) addresses. In addition, if the source and destination PAN identifiers are the same, one of them can be elided. Hence, this field containing the source and destination addresses as well as the source and destination PAN identifiers, has variable length, and it is to be interpreted according to the FCF.
Data Payload	The data being transmitted.
FCS	Frame Check Sequence for data integrity verification.

2.1.4 Internet Protocol

The Internet Protocol (IP) is the principal Internet "Network" Layer protocol. It is a connectionless, best-effort, unreliable inter-networking protocol which provides the necessary functions to deliver a packet from a source to a destination (both identified by fixed

length addresses) over a system composed of an arbitrary number of networks. It also provides mechanisms for packet fragmentation and reassembly, if needed.

The Internet Protocol was first defined by Vint Cerf and Robert Kahn in an IEEE journal paper entitled "A Protocol for Packet Network Interconnection" [13]. The protocol was later revised and updated up to its fourth version (IPv4), which is defined in RFC 791 [14], and became the first widely deployed version of IP.

2.1.4.1 IPv4

Internet Protocol version 4 (IPv4) is defined in RFC 791 [14] (replacing its previous definition in RFC 760 [15]). It uses 32-bit addresses, which limits the total number of IPv4 addresses to 2^{32}. Its header has a variable length (due to the options field), as illustrated in Figure 2.4 and described below. These two features (address length and variable length), together with the need for Flow Labeling capability constitute the main shortcomings/limitations of the protocol, and hence the reasons that have made necessary the definition of its next version (version 6).

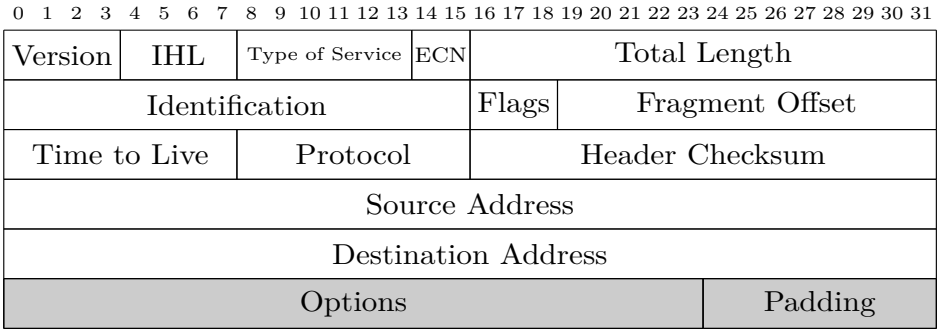


Figure 2.4: IPv4 datagram header.

The Fields of the IPv4 frame are explained next.

Version Internet Protocol version. It has a value of 4 for IPv4.

IHL Internet Header Length in multiples of 4 bytes. It is required since the header may contain a variable number of options.

Type of Service The Type of Service (ToS) field provides an indication of the parameters of the quality of service desired. It is used to specify the treatment of the datagram during its transmission. RFC 2474 redefines this field as the "Differentiated Services field" (DS field) due to the limited practical use of the Type of Service field and the need for a new field by new real-time protocols.

ECN Explicit Congestion Notification (formerly part of ToS).

Total Length The total length of the packet, including the variable length header. This field is needed to calculate the payload length, and imposes a maximum total packet length of $2^{16} - 1 = 65535$ bytes.

Identification Numeric identifier used to uniquely identify a set of fragments belonging to the same packet.

Flags Used for fragmentation control, indicating whether a fragment is the last fragment of a packet, or if fragmentation is allowed for a packet.

Fragment Offset Specifies the offset of a fragment relative to the beginning of the original packet. This field is required for packet reassembly.

Time to Live Sets a maximum packet lifetime, to prevent packets from persisting in the network due to, for example, routing loops.

Protocol Indicates the protocol of the packet encapsulated by the IP header and transported in the IP payload.

Header Checksum	16-bit checksum field, used for header error-checking.
Source Address	32-bit IP address of the source of the datagram.
Destination Address	32-bit IP address of the destination of the datagram.
Options	Optional field. It can contain a list of different options, but it must always be terminated with an "End of Options" option.
Padding	Since the number of options is variable and the length of each option is also variable, and the header length field (IHL) is expressed in 32-bit multiples, padding is needed to ensure that the header contains an integral number of 32-bit words.

2.1.4.2 IPv6

The IP protocol version 6 (IPv6) is defined in RFC 2460 [16] (replacing its previous definition in RFC 1883 [17]). It was defined in 1998 in order to succeed IPv4, with the goal of overcoming a number of IPv4 shortcomings, especially, for dealing with the anticipated IPv4 address exhaustion.

The primary changes from IPv4 to IPv6 are an increased address space, which is 128 bits (allowing for up to 2^{128} which is about 3.4×10^{38} different IPv6 addresses), a simplified header format, which includes a fixed header-length and improved support for extension headers and options (allowing for more efficient packet forwarding and greater flexibility for introducing new options), flow labeling capability (with which the sender is allowed to request special handling by routers), and authentication and privacy capabilities. The IPv6 header format is described below and illustrated in Figure 2.5.

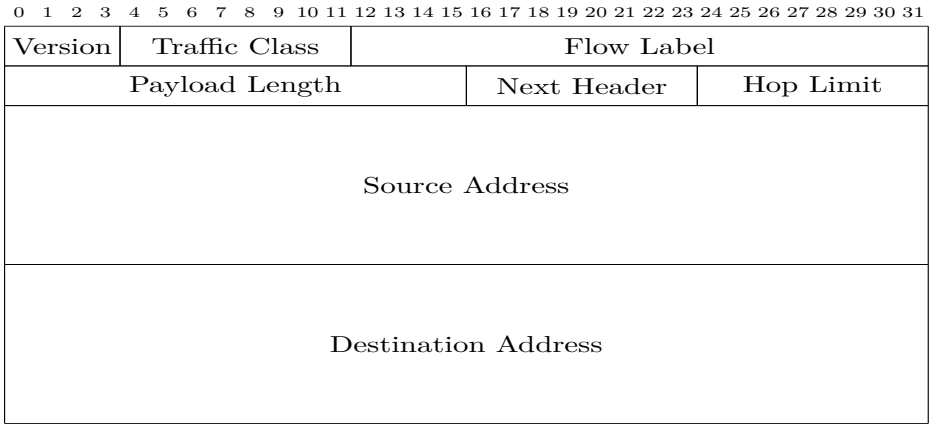


Figure 2.5: IPv6 datagram header.

Next, we explain the fields in more details.

Version Internet Protocol version. It has a value of 6 for IPv6.

Traffic Class Identifies different priorities.

Flow Label Used by the source to label sequences of packets for which it requires special handling by routers, such as a non-default quality of service or "real-time" service. This field replaces the "Type of Service" field in IPv4.

Payload Length Length of the IPv6 packet payload, not including the length of the header (40-bytes fixed length). Note that extension headers, if present, are considered part of the payload.

Next Header Identifies the type of header immediately following the IPv6 header. This next header may indicate any upper-layer protocol or an IPv6 extension header.

Hop Limit	Packet lifetime. Used to prevents packets from indefinitely persisting in the network. It is specified as a number of hops (in contrast to the "Time to Live" IPv4 field, which is specified in seconds, requiring nodes to perform difficult time computations), which is decremented at every node where the packet is forwarded.
Source Address	128-bit IP address of the source of the datagram.
Destination Address	128-bit IP address of the destination of the datagram.

2.1.5 The Internet of Things

The Internet of Things [18] is a paradigm which aims to provide everyday objects with a unique address, enabling their integration into the Internet. These objects are expected to provide contextual information and/or perform certain actions, according to their own interpretation of context and/or the orders received from remote hosts. This fact makes IPv6 (especially, because of its extremely large address space) a perfectly suited protocol for its use in the identification and communication between these objects and the rest of the Internet.

It is important to note that these objects do not require special capabilities: since a unique bar code or unique identifier in a radio-frequency identification (RFID) tag is sufficient to provide a unique identifier to an object, enabling every object to be identified and hence, integrated into the the Internet. However, the more processing capabilities the object has, the wider its communication capabilities will be; while some of these objects may have a read-only RFID tag (which may inform others of the object's location when passing RFID readers located in known places), other "things" might implement a fully-compliant IPv6 protocol stack, becoming "first-class Internet citizens" capable of sending

and receiving information to and from the Internet, just as any other network attached computer might. Between these two extremes, a vast range of possibilities exists, in which each object is required to implement only the minimum features necessary for its specific application.

Therefore, the Internet of Things concept provides for a large set of applications such as home automation, security, monitoring, smart metering, and management among others, providing the means to transform their environments into a smart, context-aware entity with the ability to sense and act. Consequently, these applications target many different markets, from individuals or families to industry.

2.1.6 6LoWPAN

6LoWPAN is an intermediate layer that allows the transport of IPv6 packets over IEEE 802.15.4 frames. Although the term 6LoWPAN stands for IPv6 over Low-power Wireless Personal Area Networks, it may extend the personal operating space, similar to LR-WPANs. RFC 4919 [30] describes an overview, assumptions, problem statement, and goals of the standard, while RFC 4944 [33] defines the standard itself. Figure 2.6 depicts how an IPv6 packet is encapsulated into a IEEE 802.15.4 frame using the 6LoWPAN adaptation layer.

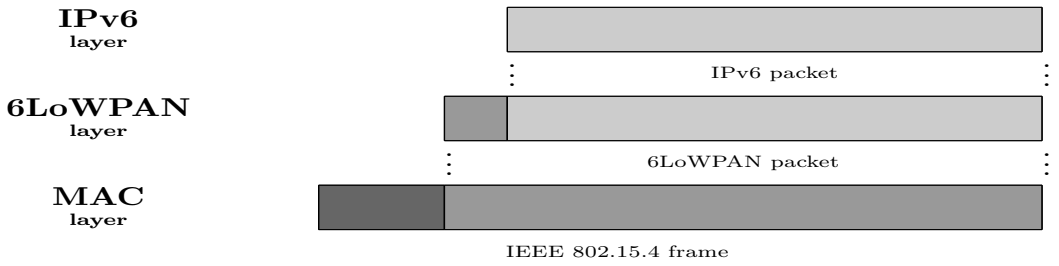


Figure 2.6: 6LoWPAN intermediate layer.

The IPv6 standard defines certain requirements for the link-layers over which it is to be transported. However, the IEEE 802.15.4 MAC layer does not fulfill these requirements in certain points. Hence, the 6LoWPAN specification defines not only the frame format for the transmission of IPv6 packets over IEEE 802.15.4, but also the mechanisms to obtain a unique IPv6 address from either the 16-bit or 64-bit IEEE 802.15.4 MAC addresses

(using Stateless Address Auto-configuration defined in RFC 4862 [19]), and to overcome the limitations of IEEE 802.15.4 [12].

2.1.6.1 Motivations

The minimum Maximum Transfer Unit (MTU) required for a link-layer transporting IPv6 packets is 1280 octets. This is far greater than the maximum IEEE 802.15.4 frame size, which is 127 octets. Of these 127 octets, the maximum MAC header size is 25 octets and, if IEEE 802.15.4 link-layer security is enabled, it may use up to 21 additional octets. This leaves only 81 octets available for IPv6 data. As the IPv6 header length is 40 bytes, only 41 bytes are available for transport layers.

In order to meet the IPv6 minimum MTU requirements, 6LoWPAN defines a fragmentation and reassembly mechanism that allows splitting IPv6 packets at the 6LoWPAN adaptation layer into smaller fragments that can be handled by the link-layer, with this process being transparent to the Internet Layer. However, applications using 6LoWPAN are not expected to use large packets. And so, in order to avoid fragmentation as much as possible, 6LoWPAN defines an IPv6 header compression mechanism which is explained next.

2.1.6.2 Packet Compression

Although RFC 4944 defines a powerful stateless compression mechanism, it can only reach its maximum effectiveness in link-local packet transmissions. This approach is inefficient for most practical cases, where 6LoWPAN devices communicate with devices external to the 6LoWPAN using routable addresses, hence the header compression mechanism defined in RFC 4944 is in the process of being updated by IP header compression (IPHC), a widely accepted new and optimized context based compression, defined in RFC 6282 [20]. In addition, there are some approaches to transport-layer header compression (still works in progress), such as the specification in [21], which describes mechanisms for generic header compression.

The IPHC compression mechanism described in RFC 6282 permits compressing the 40-byte IPv6 header down to 2 octets for link-local communications and to 3 octets for non-link-local transmissions, which corresponds to compression rates of 95% and 92.5% respectively. Note that these 2 or 3 bytes include the 6LoWPAN dispatch bit field (1 octet), which would be included even if no compression at all is used (in fact, the IPHC compression mechanism makes use of the 5 rightmost bits of the 6LoWPAN dispatch bit field for compression rather than for identification). Figure 2.7 illustrates the structure of the 6LoWPAN IPHC header (as bit fields). The fields are explained in more details next.

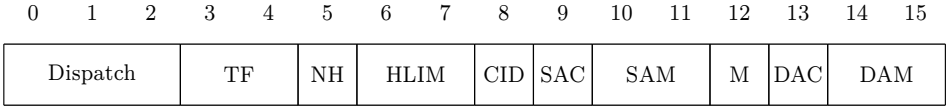


Figure 2.7: 6LoWPN IPHC base header.

Dispatch 6LoWPAN Dispatch value for IPHC compression, has a constant value of 011_2.

TF Traffic Class and Flow Label; this field being 11_2 means that both have a value of 0 and are elided.

NH Specifies whether the Next Header is carried in-line or compressed.

HLIM Hop-Limit.

CID If set, an 8-bit field containing context identifiers follows this header.

SAC Indicates whether the Source Address Compression is stateless or stateful.

SAM Indicates the Source Address compression Mode.

M If set, destination address is Multicast.

DAC Indicates whether the Destination Address Compression is stateless or stateful.

DAM Indicates the Destination Address compression Mode.

As previously mentioned, this compression mechanism relies on shared context, which means that mechanisms to maintain and disseminate such contexts must be provided. Although RFC 6282 does not specify which information a context is composed of, or how maintenance and dissemination of contexts are performed, these features have been further defined in [22].

2.1.6.3 Fragmentation

In order to comply with the IPv6 specification [16], 6LoWPAN provides support for packet fragmentation. When an entire IPv6 datagram does not fit within a single IEEE 802.15.4 MAC frame, such datagram should be split into fragments. Each of these fragments needs to be encapsulated into a 6LoWPAN packet adding a fragmentation header which specifies an IPv6 packet identifier (so each fragment can be associated with its corresponding IPv6 datagram), an offset, and the total IPv6 packet length. This fragmentation header adds a 4 octet overhead for the first fragment, and 5 for the second and subsequent fragments. Therefore, in order to minimize this overhead, fragmentation should be avoided as much as possible.

2.1.7 Contiki Operating System

The Contiki operating system is an open source operating system for networked embedded systems in general, and smart objects in particular. The first version of Contiki was released in 2003. It is developed by a team of developers from the industry and academia. The Contiki project is lead by Adam Dunkels of the Networked Embedded Systems Group at the Swedish Institute of Computer Science (SICS), and since then several other developers worked on the OS to provide it several new features.

Contiki provides mechanisms that assist the programmer when developing software for smart object applications as well as communication mechanisms that allow smart objects to communicate with each other and the surrounding world. Contiki provides libraries for memory allocation and linked list manipulation as well as communication abstractions and low-power radio networking mechanisms [23]. Contiki has a file system called Coffee that allows programs to use flash ROMs as a traditional file store [24]. Additionally, Contiki provides a set of simulators that simplify the development and experimentation with smart object networks [25, 26].

Contiki was the first operating system for smart objects that provided IP communication with the uIP TCP/IP stack [27, 28]. In 2008, the Contiki system incorporated uIPv6, the world's smallest IPv6 stack [29]. The footprints of the uIP and uIPv6 stacks are small: less than 5 kB for the uIP stack and approximately 11 kB for uIPv6. This makes them suitable for use in the constrained environment of a sensor or a smart object.

Many components of Contiki are widely used in the industry. The uIP TCP/IP stack, and its larger cousin lwIP, is currently used by hundreds of companies in products ranging from car engines and airplanes to worldwide freighter container tracking systems and satellite systems. The protothread programming abstraction used in Contiki [30] is used in systems such as digital TV set-top boxes and high-performance server clusters.

Both the Contiki system and applications for the system are implemented in the C programming language. Because Contiki is implemented in C, it is highly portable. Contiki has been ported to more than twelve different microprocessor and microcontroller architectures.

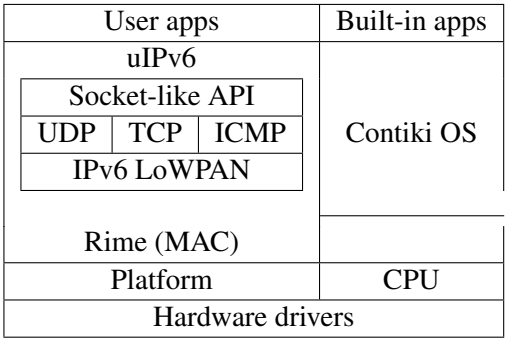


Figure 2.8: The architecture of Contiki.

The Contiki OS general features are as follows:

- Full IPv4 and IPv6 support for IP communication, using the uIPv6 stack.
- Multitasking kernel, with support for multithreading programming using pre-emptive multithreading and protothreads.
- Power efficient radio and network mechanisms, 6LoWPAN header compression, RPL routing and CoAP application layer protocol.
- Applications such as an HTTP server and Telnet client.
- Supports several simulators, such as Cooja and MSPSIM, to aid in the software development and debugging process.
- A proprietary file system for data storage.

The 6LoWPAN implementation for Contiki OS is called SICSlowPAN, being based on RFC 4944, as well as "Interoperability Test for 6LoWPAN" [31], and "Compression format for IPv6 datagrams in 6lowpan Networks" [32]. SICSlowPAN is an adaptation layer mechanism. When a Contiki device receives an IPv6 packet, the MAC layer calls SICSlowPAN to adapt the packets to be used by the IPv6 layer (being implemented by the uIPv6 stack) and when uIPv6 needs to send an IPv6 packet also calls SICSlowPAN to adapt it for the IEEE 802.15.4 standard MAC frames.

2.1.8 TinyOS

TinyOS is another free and open-source operating system developed for embedded systems with memory-constrained devices, such as IEEE 802.15.4 network motes. Unlike Contiki, TinyOS has no multi-threading capabilities, being an event-driven architecture. TinyOS is implemented in NesC, a different programming language based in C, which limits the portability of the operating system.

The first implementation of 6LoWPAN for TinyOS was called 6lowpancli, featuring the HC1 and HC2 header compression, addressing and fragmentation, IPv6 stateless configuration and ICMPv6 support. Later Berkeley, from University of California released their implementation of 6LoWPAN - b6lowpan, usually called blip. Blip is more than an implementation, it is an IPv6 stack including Neighbor Discovery, support for TCP, UDP, DHCPv6, has a point-to-point daemon to communicate with Unix machines and is the basis for the TinyRPL and CoAP implementations. The 6LoWPAN implementation is based on [32]. It also includes both mesh under and route over mechanisms.

Due to the ease in programing with ContikiOs as well as its portability. We have decided to work with it to perform the simulation and experimentation throughout this thesis.

2.1.9 COOJA

COOJA is a flexible Java-based simulator designed for simulating networks of sensors running the Contiki operating system [33]. COOJA simulates networks of sensor nodes where each node can be of a different type; differing not only in on-board software, but also in the simulated hardware. COOJA is flexible in that many parts of the simulator can be easily replaced or extended with additional functionality. Example parts that can be extended include the simulated radio medium, simulated node hardware, and plug-ins for simulated input/output.

A simulated node in COOJA has three basic properties: its data memory, the node type, and its hardware peripherals. The node type may be shared between several nodes and determines properties common to all these nodes. For example, nodes of the same

type run the same program code on the same simulated hardware peripherals. And nodes of the same type are initialized with the same data memory. During execution, however, nodes' data memories will eventually differ due to, e.g., different external inputs.

COOJA currently is able to execute Contiki programs in two different ways. Either by running the program code as compiled native code directly on the host CPU, or by running compiled program code in an instruction-level TI MSP430 emulator. COOJA is also able to simulate non-Contiki nodes, such as nodes implemented in Java or even nodes running another operating system. All different approaches have advantages as well as disadvantages. Javabased nodes enable much faster simulations but do not run deployable code. Hence, they are useful for the development of, e.g., distributed algorithms. Emulating nodes provides more fine-grained execution details compared to Javabased nodes or nodes running native code. Finally, native code simulations are more efficient than node emulations and still simulate deployable code. Since the need of abstraction in a heterogeneous simulated network may differ between the different simulated nodes, there are advantages in combining several different abstraction level in one simulation. For example, in a large simulated network a few nodes may be simulated at the hardware level while the rest are implemented at the pure Java level. This approach combines the advantages of the different levels. The simulation is faster than when emulating all nodes, but at the same time it enables a user to receive fine-grained execution details from the few emulated nodes.

COOJA executes native code by making Java Native Interface (JNI) calls from the Java environment to a compiled Contiki system. The Contiki system consists of the entire Contiki core, pre-selected user processes, and a set of special simulation glue drivers. This enables the deployment and simulation of the same code without any modifications, minimizing the delay between simulation and deployment.

The Java simulator has full control over the memory of simulated nodes. Hence the simulator may at anytime view or change Contiki process variables, enabling very dynamic interaction possibilities from the simulator. Another interesting consequence of using JNI is the ability to debug Contiki code using any regular debugger, such as gdb, by attaching it to the entire Java simulator and breaking when the JNI call is performed.

Also entire simulation states may be saved and later restored, skipping back simulations over time.

The hardware peripherals of simulated nodes are called interfaces, and enable the Java simulator to detect and trigger events such as incoming radio traffic or a LED being lit. Interfaces also represent properties of simulated nodes such as positions that the actual node is not aware of.

All interactions with simulations and simulated nodes are performed via plugins. An example of a plugin is a simulation control that enables a user to start or pause a simulation. Both interfaces and plugins can easily be added to the simulator, enabling users to quickly add custom functionality for specific simulations.

Each simulation in COOJA uses a radio model that characterizes radio wave propagation. New radio models may be added to the simulation environment. The radio model is chosen when a simulation is created. This enables a user to, for example, develop a network protocol using a simple radio model, and then testing it using a more realistic model, or even a custom made model to test the protocol in very specific network conditions. Often a radio model provides one or several plugins in order to configure and view the current simulated network conditions.

COOJA supports, except from a completely silent model, a simple model that uses an interference and a transmission range parameter that can be changed during a simulation run. Ongoing work on better radio models will provide COOJA with a general ray-tracing based model supporting radio absorbing material.

2.1.10 TelosB Sky Motes

The Crossbow TelosB is an open-source platform developed by University of California, Berkeley [34]. It provides a very power efficient sensor mote, equipped with the Texas Instruments MSP430 micro-processor, the IEEE 802.15.4 compliant Texas Instruments CC2420 RF chip, 8 ADC channels for sensing data, sensors for temperature, light and humidity, as well as several interface ports. USB support is also provided, for an easier programming environment or communication with a PC. The TelosB is an architecture

that is widely supported for both TinyOS and Contiki OS applications, as well as the Cooja simulator.

There are other platform available. However, we have found that the Sky motes are the platform that suits the need of our work the best.

Advanticsys MTM-CM5000-MSP shown in Figures 2.9 and 2.10 is one of the TelosB architecture products available on the market, being the platform chosen to carry on the tests on this thesis. Table 2.2 states the technical specifications of the product.

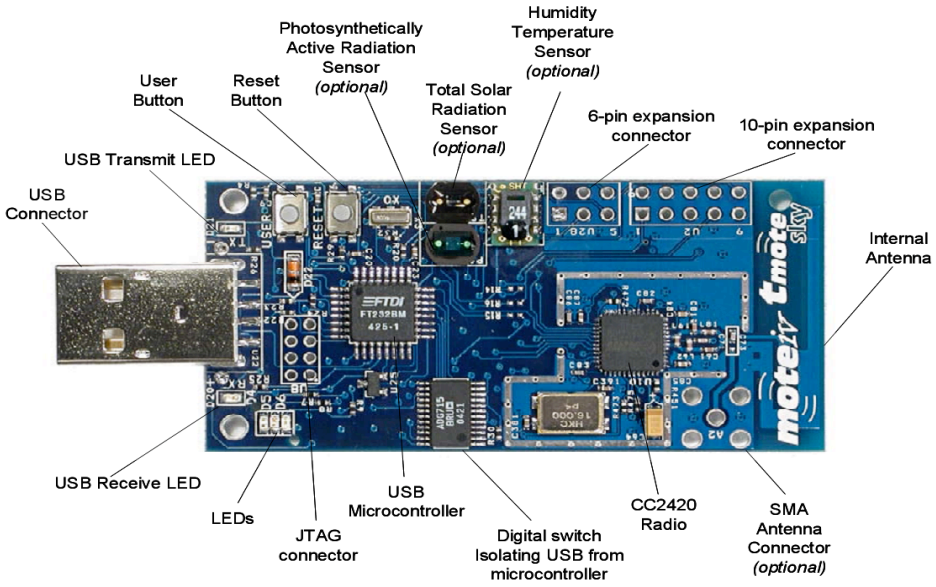


Figure 2.9: Tmote Sky front.

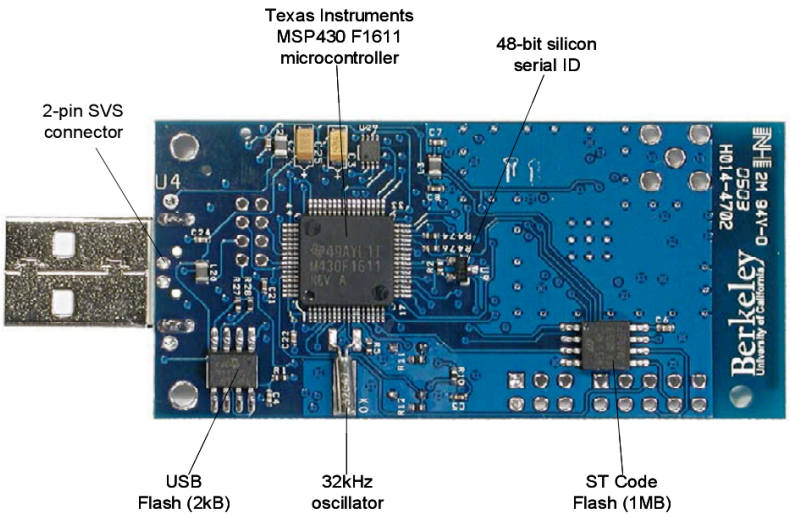


Figure 2.10: Tmote Sky back.

Table 2.2: Advanticsys MTM-CM5000-MSP general specifications.

Processor	Texas Instruments MSP430F1611
Memory	48KB ROM 10KB Data RAM 1MB External flash memory
ADC	8 channels with 12bit resolution
Interfaces	USB, UART, SPI, I2C
RF Chip	Texas Instruments CC2420
Frequency Band	2.4GHz 2.485GHz
Sensitivity	-95dBm
Transfer Rate	250Kbps
RF Power	-25dBm 0dBm
Range	120m(outdoor), 20 to 30m(indoor)
RF Chip Current Draw	RX: 18.8mA TX: 17.4mA Sleep mode: 1uA
Included Sensors	Hamamatsu S1087 Light Sensor Sensirion SHT11 Temperature and Humidity sensor
Dimensions	81.90mm x 32.50mm x 6.55mm
Weight	17.7g (without batteries)

2.2 Related Work

In this section, we list some of the work found in literature that relates to the work we have done within the scope of this thesis. We have tried to summarize the important papers that were published which were concerned with topics similar to the ones we are addressing in this thesis.

These papers are divided into two groups, each related to one of the main contributions of this thesis. The Self-Configuration of nodes and the Data Collection from nodes.

2.2.1 Self-Configuration

Self-configuration in large WSNs has been a hot research topic in the last years. There are many examples in the literature that require deploying wireless sensors that can form networks to make and convey measurements for many applications. For example, measuring ocean temperatures and currents, analyzing moisture content in soils, assessing sunlight in forests, and monitoring stresses in structural supports of large buildings and bridges. The number of devices, communications channels, and data transmissions will become too large, varying, and uncertain to be deployed and managed with the costly techniques in use today.

Instead, wireless networks must become adept at self-organization—allowing devices to reconnoiter their surroundings, cooperate to form topologies, and monitor and adapt to environmental changes, all without human intervention.

Self-organization applied to wireless networks is not a new concept. In this subsection, we will go over the main papers that we thought relate to the subject and to our work.

In [35], the author introduced the current, scientific understanding of self-organizing systems and identified the main models investigated by computer scientists in literature seeking to apply self-organization to design large, distributed systems. Additionally, the author has made an intensive survey on the research that uses models of self-organization in wireless sensor networks to provide a variety of functions. Such functions presented include: sharing processing and communication capacity, forming and maintaining structures, conserving power, synchronizing time, configuring software components, adapting

behavior associated with routing, with disseminating and querying for information, and with allocating tasks, and providing resilience by repairing faults and resisting attacks. The paper closed with a summary of open issues that must be addressed before self-organization can be applied routinely during design and deployment of sesnor networks and other distributed, computer systems. This paper gave us a good base to stand on when addressing the objective of this thesis.

In [36], the authors proposed a mesh radio based solution which exhibits self-organizing characteristics and claimed that therefore it is attractive from a deployment perspective. The solution they proposed was created as an enhancement to the RPL routing protocol for low power and lossy networks. Their solution enables smart meter nodes augmented with the mesh radio to automatically discover sink nodes in the vicinity, setup a single/multi-hop link to the best available sink, detect loss of connectivity and subsequently re-configure itself to re-establish connectivity so as to ensure reliable transport of smart metering data. Moreover, it achieves this in a distributed manner thereby ensuring scalability. They have evaluated the performance of their proposed solution over different scenarios with different node distributions. We have followed a similar approach in our work for the self-configuration and identification of neighbors. However, since we are not using RPL, we had to create our solution based on a lower layer approach.

In [37], Lee et al. have proposed a self-organized and smart-adaptive clustering (SOSAC) and routing method, which performs clustering in WSNs, operates the formed clusters in a smart-adaptive way, and performs cluster-based routing. It can be referred to as a hierarchical clustering and routing model capable of maximizing the network lifetime through the decision making of each sensor node based on local information by adopting a self-organized and smart-adaptive system in the design of the clustering and routing model of a WSN. They claimed that their proposed method enables the sensor nodes to form clusters without a server or any external assistance, and the subsequent routing is performed based on it. The key advantage of this model is its ability to sense any environmental disturbances, such as time changes, number of sensor nodes, and failures of sensor nodes, using three smart adaptive mechanisms. The authors have compared the performance of the proposed SOSAC with that of a well-known clustering and routing protocol

for WSNs. They have shown by computational experiments that the network lifetime, energy consumption, and scalability of SOSAC are better than those of the compared method. The research done in this paper intended mainly to reduce the power consumption of the nodes and to increase the lifetime of the network. Since our work is based on a power plant environment, we have no power limitations, and so this research did not fit the needs of this PhD work.

In [38], Patrawi et al. have proposed a sensor localization system based on either the Received Signal Strength (RSS), Quantized RSS, or the proximity measurements between the sensors. They have showed analytically that the standard deviation bound for proximity measurements was 48% worse than that for RSS measurements. Additionally, when the nodes are placed in a grid setup, proximity measurements had an average standard deviation bound about 57% worse than that of RSS measurements. As for the K-level quantized RSS, they have shown that it performs as well as RSS, even for low values of K. In our work, we use the value of the Received Signal Strength Indicator (RSSI) that is measured by the radio driver of the sensor node which is similar to the QRSS value mentioned in that paper, but with no quantization.

In [39], the authors investigated the use of Radio Frequency (RF) location systems for indoor domestic applications. They introduced the concept of RF location system for Integrated Indoor Location Using RSSI and Link Quality Indicator (LQI) provided by the ZigBee module. They have shown different ideas to overcome the problems in the existing methods for calculating the distance in indoor environments. They have presented a new method for reducing the error in the location identification due to interference within the infrastructure-based sensor network. The proposed method calculates the distance using the LQI and the predicted RSSI based on the previously measured values. The calculated distance corrects the error induced by interference. They have shown by experimental results that their method can reduce the average error around 25%, and it is always better than the other existing interference avoidance algorithms. In our work, we care more about the relative location of nodes with respect to the other nodes in the network, instead of estimating the real distance between nodes.

In [4], the authors have proposed what they called an Efficient Self-Organization Algorithm for Clustering (ESAC), which they based on the weighing parameters k-density, residual energy of the nodes and node mobility for cluster heads election. They showed that their algorithm enables the creation of low number of stable and balanced clusters while avoiding the problem of battery drainage of the cluster head. They have proven that this algorithm prolongs the lifetime of the network while reducing the broadcast overhead in the network. We also are interested in the operation of the election of cluster head. However, we have used the relative location of the node to be the main factor in that process.

In [40], Borbash et al. presented an asynchronous and distributed algorithm for neighbor discovery. The proposed algorithm was shown to allow each node in the network to populate a neighbor list that may not be complete. Their algorithm is set to run at boot time and can be re-run whenever the network changes. In this algorithm, each node has its own time slotting and an initial estimate on how many neighbors it should have. In each time slot, nodes alternate between transmitting and receiving states with different probabilities and try to listen for messages sent from the other nodes. Each node runs independently of the other nodes and adds a node to its neighbors list as soon as it receives a message from it. The authors have analyzed the performance of their algorithm and shown that they can tune it to maximize the percentage of the neighbors discovered in a fixed running time. We are doing a similar job in this paper. However, we do not allocate specific time slots per node. We rely on the randomness in the message sending as well as a sufficiently large number of messages to be sent in order to assure that each node hears all of its neighbors. Additionally, we use RSSI to differentiate between one hop neighbors and further away nodes.

2.2.2 Data Collection

In dense wireless sensor networks, it is often necessary to collect the data accumulated by each sensor node for processing at a central location. The operation of transmitting accumulated data from sensor nodes to the sinks is called data collection. As the network

grows, the need arise to schedule the collection of a large amount of data from sensor nodes to sinks. Normally, data need to be collected without merging. However, if the data can be merged, the operation is then called data aggregation. One of the challenges in data collection in wireless networks is radio interferences that may prevent nearby sensor nodes from transmitting packets simultaneously. Scheduling data transmissions without carefully considering such inferences can result in significant delay in data collection due to the collisions and retransmissions. Below is an overview of some of the papers that discuss the problem of data collection from nodes, with and without data aggregation, that were found to be relevant to our work.

In [41], Wang et al. presented an in-depth survey on recent advances in networked wireless sensor data collection. They highlighted the special features of sensor data collection in WSNs, by comparing it with both wired sensor data collection networks and other applications using WSNs. In general, data collection can be broken into the deployment stage, the control message dissemination stage, and the data delivery stage. They discussed issues and prior solutions on sensor network deployment and data delivery protocols. Also, they discussed different approaches for control message dissemination, which acts as an indispensable component for network control and management and can greatly affect the overall performance of WSNs for sensor data collections. In our work, we test the performance of a WSN in a strip-based deployment while employing different data techniques that were mentioned by [41].

In [42], another comprehensive survey on data-aggregation algorithms in WSNs is presented. The authors showed that techniques for data aggregation aim to improve aspects such as network lifetime, latency, and accuracy of the sensed data. They presented an overview of many data aggregation algorithms found in literature showing their advantages and disadvantages. They showed that the performance of any data aggregation algorithm is connected to the infrastructure of the network. And so, not any protocol can perform well in every type of network. We have used a basic method for data aggregation here in a simple multi-hop network in order to reduce interference between the nodes.

In [43], the authors proposed a scheme for Adaptive Packet Concatenation (APC). In this scheme, several data link layer packets destined to the same node are concatenated

in a super packet that is sent once. This reduces protocol overhead coming from sending multiple packets to the same destination. They have proved analytically that this scheme can improve the throughput of the network from 4 to 16 times. We are doing something similar. However, their work was done in an IEEE 802.11 network performing the aggregation at the data link layer, while we are using IEEE 802.15.4 MAC protocol and we are aggregating data on the application layer.

In [44], Patel et al. have studied query processing as well as data aggregation in wireless sensor networks. They have showed that optimizing data aggregation in a WSN can help with the reduction of energy consumption in the network and in turn increase the life span of the sensors' battery and remove any redundancy in data transmission. They have used TinyOs and its simulation tool TOSSIM to simulate a multi-hop sensor network. They have also used a multi-hop network in order to reduce both transmission and receiving powers, reducing the overall power consumption as well as the interference between nodes. Our work here follows a similar path. However, we focused more on data throughput and less on energy consumption.

In [45], the authors proposed a collision-free data aggregation scheduling algorithm. In this algorithm, each sensor node of the same parent node is assigned a consecutive time slot in order to reduce the frequency of state transitions. In this way, the parent node can only start up once to receive all the data from its children. This reduces energy consumption of the nodes as well as remove latency coming from state transitions. They have showed by theoretical analysis and simulations that their algorithm performs well in terms of number of state transitions, energy consumption, and time delay. In our work, we do not assign specific time slots for each node. However, each node waits until it hears from its neighbor before sending. This is how we avoid collisions without adding more management tasks to the nodes, such as the time slot assignments.

In [46], Accettura et al. evaluated the performance of Routing Protocol for Low Power and Lossy Networks (RPL) [47], using the Contiki COOJA simulator. They showed that RPL can grant very fast network set-ups and bounded communication delays. Additionally, its effectiveness can be further improved in terms of overhead, which can be very high due to Destination Advertisement Object (DAO) messages, used to handle sink-to-

node downward traffic. We use RPL in our work with storing mode and DAO messages in order to allow downward traffic from the sink to the rest of the nodes.

In [48], Wang et al. proposed an RPL based routing protocol design for Advanced Metering Infrastructure (AMI) networks in smart grid. They used expected transmission count (ETX) as the link metric and proposed a low-cost ETX measurement scheme. Also, the authors have proposed a novel, ETX-based rank computation method to be used for the construction and maintenance of the RPL routing tables, which provided high end-to-end reliability for the inward unicast traffic in AMI networks. Additionally, they proposed a reverse path recording mechanism that establishes the paths for the outward unicast traffic in AMI networks. The simulation results presented in that paper showed that, in the presence of shadow fading, their proposed routing protocol produces satisfactory performances in terms of packet delivery ratio and end-to-end delay. In our work, we assume similar configuration of nodes sensing the voltage output of the panels. However, we use the hop count metric when computing the rank.

Chapter 3

Self-Configuration

Wireless Sensor Networks (WSNs) are made of a large amount of small devices that are able to sense changes in the environment, and communicate these changes throughout the network. As the network grows, it becomes impractical and even impossible to configure all these nodes manually. And so, the use of self-organization and auto-configuration algorithms becomes essential.

The work done in this chapter aims to enable the complete self configuration of a dense wireless sensor network without any interactions from the human operators. As stated in the introduction, the communications scenario is provided by the SELF-PVP project [49]. This project aims to increase the efficiency of a photo voltaic (PV) power plant. It assumes that 200,000 solar panels are deployed in a matrix like deployment spread over a large area (250 hectares). The goal of the project is to build a self-organizing, truly distributed computational network reflecting ambient intelligent, with the ability to sense and control the operation point of each panel in a PV power plant, in order to accomplish a global maximum power available at any environment condition of operation. Each solar panel will have a sensor/actuator that will sense local variables, communicate these values with other sensors and compute local errors with the goal of optimizing the overall performance of the panels' array. While we use the SELF-PVP as the reference network to test the work done in this chapter, the work can be applied to other networks that follow the same trend. For example, a parking garage network in which sensors are also deployed in a matrix like grid. Where sensors are static and powered by local sources and required to

sense and transmit the vacancy condition of the lot to a central location. Another example is a smart energy metering system where each sensor is connected to and powered by a meter and is required to read the energy consumption and send it to a central location for accounting.

We will focus in this chapter on the example of a PV power plant where panels are spread over a huge area. A WSN in this scenario can be viewed as a grid of sensors where each sensor is connected to a solar panel. Panels in the same column will have the same value of the current passing through them. In order to reach the optimum power delivery point, most of the communications will be among nodes in the same column.

Based on this, we divide the network into small networks of columns with fixed sizes. Each column is to operate in a different frequency channel in order to reduce interference between adjacent columns. These networks can be considered as building blocks that can be replicated as many times as needed in order to cover the whole of the dense WSN. Additionally, because we are dealing with a power plant, we assume that nodes do not need to sleep in order to save energy as they can get all the energy they need from the panels.

The aim is to allow each node in the network to automatically identify its closest neighbors as well as its relative location using the value of the Received Signal Strength Indicator (RSSI) of the messages sent back and forth among nodes during the setup phase of the network. This study considers networks with different sizes, mainly by adding more columns to the network with each column having 9 nodes.

The network self-configuration was divided into three algorithms, the Neighbor Identification, the Relative Location, and the Frequency Allocation Algorithms.

In this Chapter, we propose and evaluate the three algorithms which together can enable the complete self-configuration of a large network of sensors. The performance of the three algorithms was evaluated using the Contiki COOJA simulator [26] which can be downloaded from [50]. Additionally, experiments on real testbeds were performed in order to validate the performance of the algorithms in different scenarios.

The rest of the chapter is organized as follows. Section 3.1 presents the topology proposed for the WSN, as well as the three self-configuration algorithms. Section 3.2

describes the simulation environment and different parameters tested in order to obtain the simulation results. These results are presented in Section 3.3. Section 3.4 describes a set of testbed experiments performed and the results obtained from running them. Finally, conclusions are listed in Section 3.5.

3.1 Proposed Topology and Self-Configuration Algorithms

3.1.1 Network Topology

As the network we are using for this study is based on a photo voltaic power plant where panels are spread over a huge area, we decided to simulate the network as a grid system of sensors where each sensor is connected to a solar panel and has constant distances between it and each of its neighboring panels. Additionally, since most of the communications will be done within each column, we decided to divide the whole network into small networks of columns. Each sensor reads the output voltage of its respective panel and sends it to a central node within the column. And since the changes in the light conditions are slow by nature, it is expected that the required traffic load is going to be low.

The proposed WSN topology is shown in Figure 3.1. Such setup is referred to as a strip-based deployment [41]. It allows for full coverage of the area under consideration. In a real life scenario, nodes within a column are placed side by side with a distance of about 2 meters between each consecutive sensor, while columns have a more wider distance between them in order to allow maintenance access.

Additionally, in order to reduce the interference between columns, each column is to operate in a different frequency channel than the other columns. To enable inter-column communication, a core network will be created operating in a frequency different from all the columns. Core network nodes (solid circles in Figure 3.1) are placed between columns to enable inter-column communication, and when the need for inter-column communication arises, the respective column heads (squares in Figure 3.1) will have to switch to this core frequency and communicate through the core network. A group of columns with

their access network can be considered as a building block which can be replicated as the network grows.

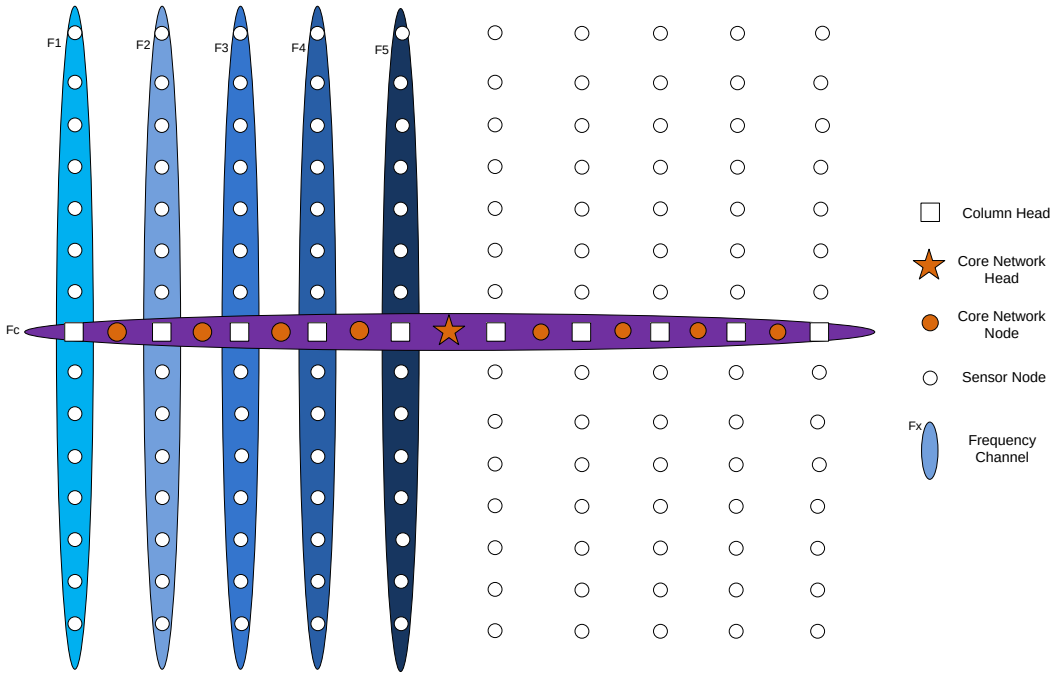


Figure 3.1: Network topology.

In this paper, three algorithms are proposed in order to perform the self-configuration of the nodes. Firstly, nodes will have to run the Neighbor Identification Algorithm in order to allow each node to find its closest neighbors, i.e., neighbors who can be reached directly without multi-hopping. Secondly, the Relative Location Algorithm will run in order to decide which node in a column is in its center and designate it a column head (squares in Figure 3.1). These column heads will then have to decide which node should be the core network head (star in Figure 3.1). And finally, the Frequency Allocation Algorithm will run. It is used to assign a different frequency channel to each of the columns in order to reduce the interference between adjacent columns. These three algorithms are described in more detail in the next three subsections.

3.1.2 Neighbor Identification Algorithm

The Neighbor Identification Algorithm is shown in Figure 3.2.

The aim of this algorithm is to allow each node in the network to find its closest neighbors, i.e., neighbors who can be reached directly without multi-hopping. Nodes on

column extremities (white circles) should have only one close neighbor. While nodes inside a column (white circles) should have two close neighbors. Column heads (squares) should have three or four close neighbors depending on the column location. And, core network nodes (solid circles and star) should have two close neighbors.

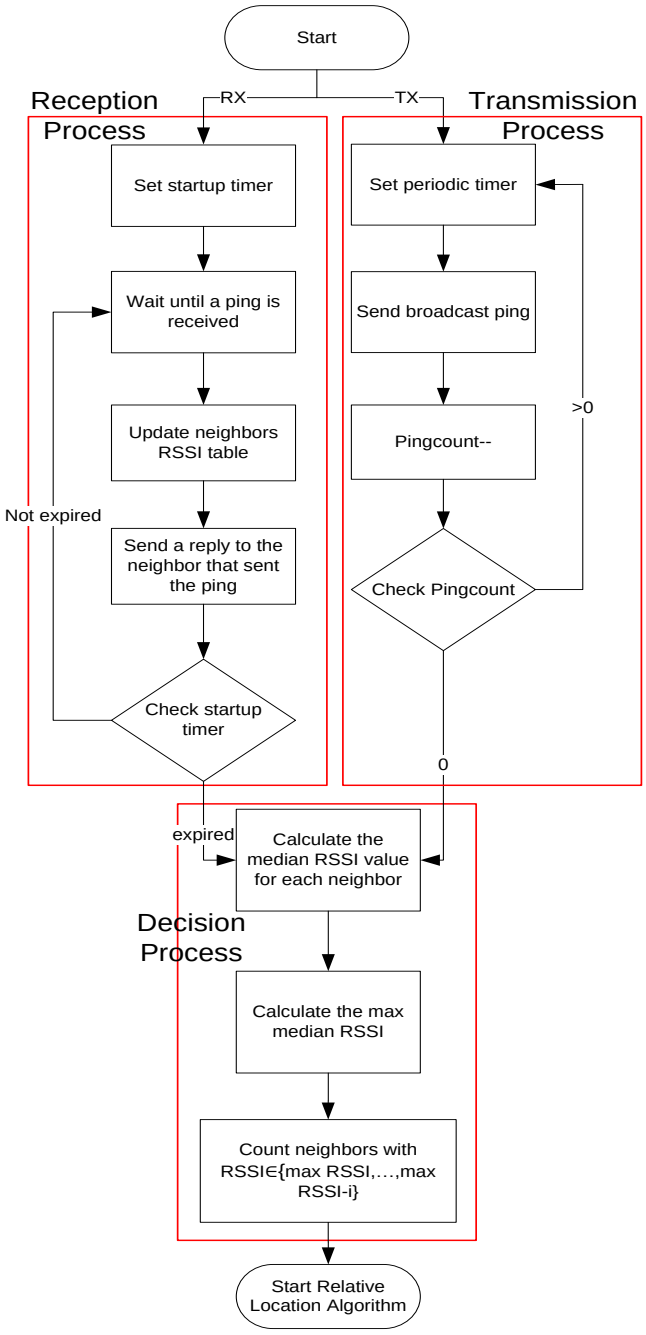


Figure 3.2: Neighbor identification algorithm.

The neighbor identification is achieved by measuring the value of the RSSI of the ping messages that are sent back and forth between nodes in the network after booting. Using

this information, nodes calculate which neighbors have the highest RSSI value. These neighbors will be considered then as the closest neighbors in the network. The algorithm runs in each node after it boots and is divided into three processes:

- **Transmission Process**:

  Each node sets a periodic timer, which controls the time period between two consecutive broadcast ping messages. Its duration is selected depending on the size of the network, i.e, the number of nodes in the network. Then, each node sends a broadcast ping message to its neighbors every time the timer expires with a small random delay added to help avoid collisions from other nodes' pings. This process is repeated until a certain number of ping messages are sent, which is predefined as a parameter of the algorithm and is referred to as the ping count.

- **Reception Process**:

  It runs simultaneously with the transmission process. A setup timer is set large enough to allow the reception of all the ping messages sent from other nodes. To ensure that, this timer should be greater than the periodic timer multiplied by the ping count. While this timer has not expired, nodes listen for ping messages sent from neighbors, and update the neighbors table with the RSSI value of the received ping of the transmitting neighbor. Then, send a reply to that neighbor so it can also update its neighbors table. This last step can be skipped and rely only on the ping messages to update the RSSI in order to reduce traffic. However, it has been found that it helps with the decision. This process is repeated until the setup timer expires.

- **Decision Process**:

  Starting only after both Transmission and Reception processes have ended, each node begins its decision making process. It starts by calculating the median RSSI value for each neighbor in its neighbors table. Then it finds which neighbor has the maximum median RSSI. Nodes then have to find how many of their neighbors have a median RSSI that lies in a set containing max RSSI and max RSSI$-i$, where i is a positive integer. This will tell the node how many close neighbors it has. The value

of i depends on the topology of the network. Initial tests has to be made in order to find the value of i that minimizes the error in the decision making. Then this value will be set in the algorithm for such a topology.

After a decision is made, the Relative Location Algorithm starts.

The algorithm uses the RSSI value of the received messages as calculated by the radio driver of the sensor node. It is a rounded value in dBm and it is mainly affected by the transmission power and the propagation medium. We have made sure that the transmission power remains constant throughout the simulation and the testbed experiments. We use the median RSSI in order to filter out any extreme values that may appear due to the noise or fading effects.

3.1.3 Relative Location Algorithm

The Relative Location Algorithm is shown in Figure 3.3. The objective of this algorithm is to enable each node to find its relative location in the network with respect to the other nodes and, based on this, decide its role in the network. After nodes have identified their closest neighbors from the output of the previous algorithm, they exchange messages among themselves in order to locate the central node of each column (the column head). Then, these column heads will also exchange different messages among themselves to try to find the central node of the network which will be referred to as the core network head (star in Figure 3.1). This algorithm begins only after the neighbor identification algorithm has ended. Each node checks the number of close neighbors it has. Based on this value, one of the following steps can be taken by the node:

- **Edge Sensor Node (one neighbor)**:

  If the node has only one close neighbor, this can only mean from the topology that it is located on the end of a column. And so, it designates itself as an **edge sensor node**, i.e., a node at the edge of a column. Then it sends a column message to its neighbor with the count 1. A column message is a control message with a specific type and a count that will be used by nodes within the same column to elect the column head.

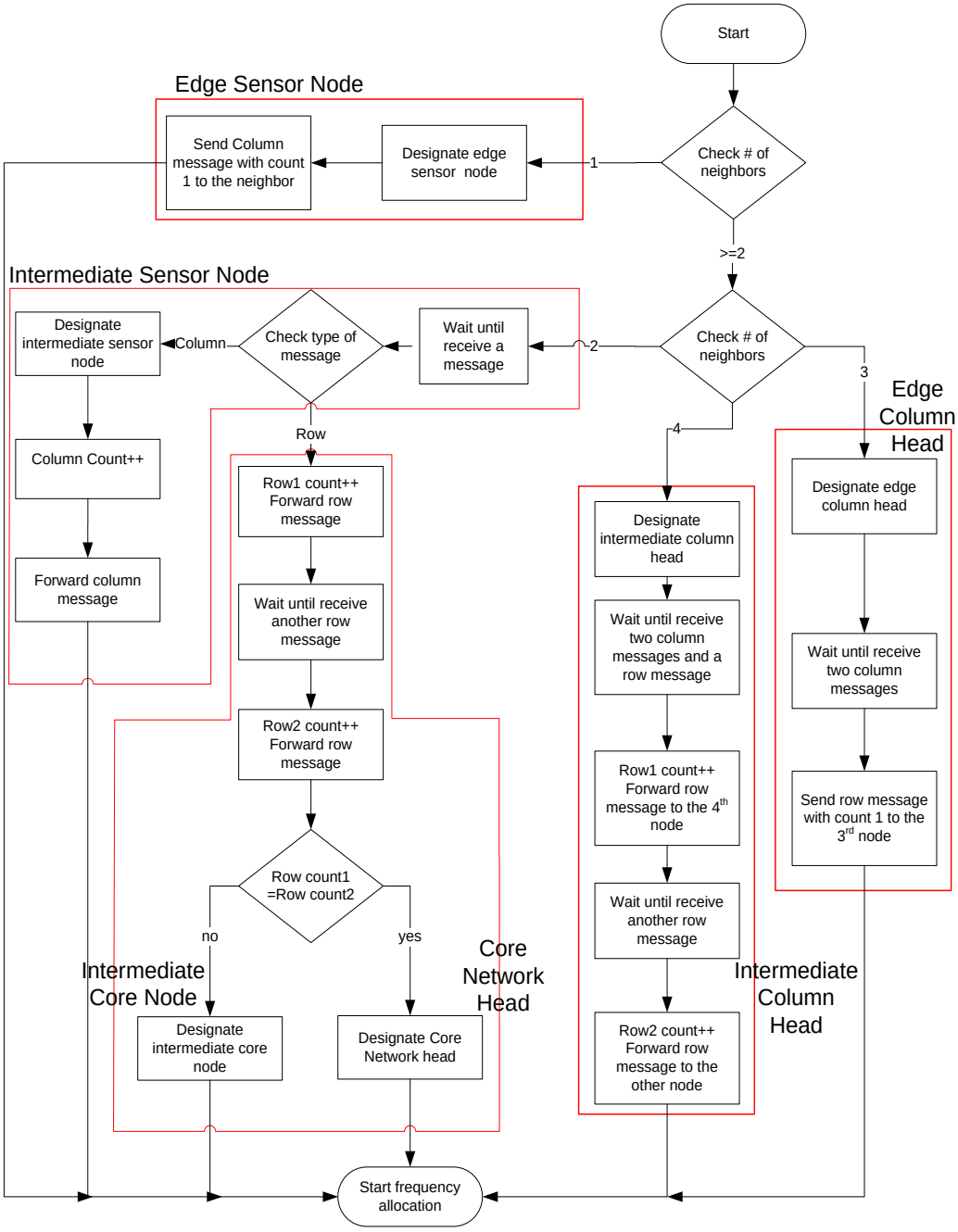


Figure 3.3: Relative location algorithm.

- **Intermediate Sensor Node (two neighbors)**:

  If the node has two neighbors, it can either be a node inside a column or a node inside the core network. And so, it waits until it receives a message. If this message is a column message, it designates itself an **intermediate sensor node**, increases the column count received from the message, and then forwards the column message with the increased count to its other neighbor in the same column.

- **Edge Column Head (three neighbors)**:

  If the node has three neighbors, it means from the topology that it can only be a node located at the edge of the core network and in the center of the edge column. And so, it designates itself as an **edge column head**, i.e., a column head at the edge of the network. Then, it waits until it receives two column messages from the two sides of its column. After receiving these two messages it confirms that it is a column head by checking that the two counts from the two column messages received are equal. Then it sends a row message to the third node with count 1. The row message is a control message with a specific type different from the column message and will be used to determine the core network head.

- **Intermediate Column Head (four neighbors)**:

  If the node has four neighbors, it can only be a node that lies inside the core network and in the center of a column. Based on this, it designates itself as an **intermediate column head**. Then, it waits until it receives two column messages from the two sides of its column, and uses this information to confirm that it is the column head as in the previous step. It then waits until it receives a row message. Then it forwards the row message to the fourth node with an increased count. Then, waits for another row message and forwards it to the other node with an increased count.

- **Intermediate Core Node / Core Network Head (two neighbors)**:

  If the node has two neighbors, and it receives a row message, it will increase the row count received from the message, and then forwards the message with the increased count to the other neighbor. It then waits for another row message and forwards it to its other neighbor with increased count.

  After receiving two row messages, the node checks the row count from both the messages. If they are not equal, it designates itself as an **intermediate core node**. If the two counts are equal, it designates itself as the **core network head**.

The core network head then begins the frequency allocation algorithm.

3.1.4 Frequency Allocation Algorithm

The Frequency Allocation Algorithm shown in Figure 3.4 is used to assign different frequency channels to each of the different columns in the network. This is done in order to reduce the interference that will happen between nodes in different columns.

When the Relative Location Algorithm ends, each node will check its type as decided by that algorithm. And according to that type, one of the following steps will be taken:

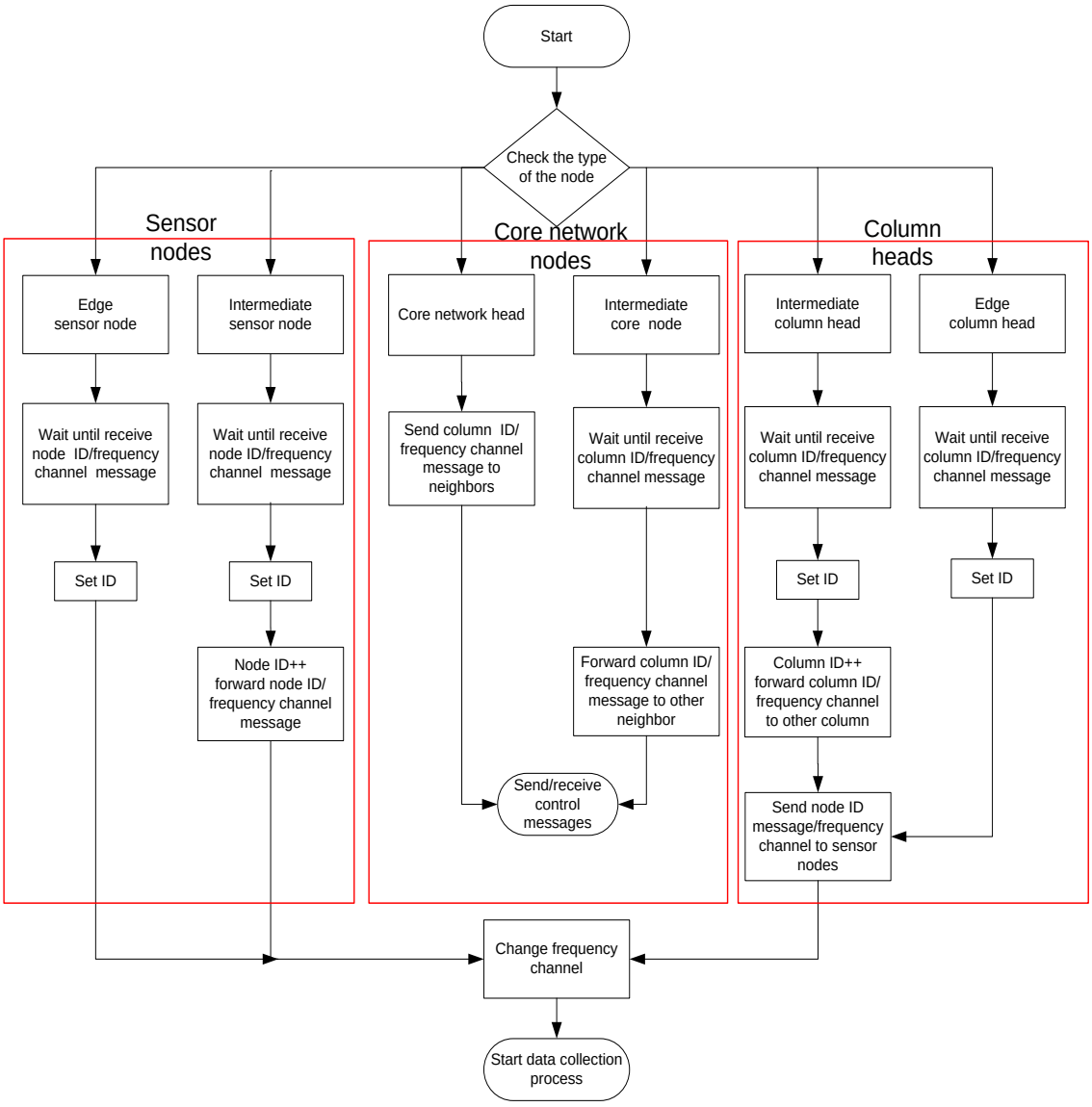


Figure 3.4: Frequency allocation algorithm.

- **Core Network Head**:

  The core network head decides how many frequency channels are needed in the network according to the number of columns. And then, it sends massages to both sides of the core network. These messages will hold the frequency channel and the column ID of the first columns on both sides of the network around the core network head. It then can be used to send or receive any control messages between the operator and the sensor nodes if needed.

- **Intermediate Core Nodes**:

  Intermediate core nodes receive the column ID/frequency channel messages and forward them to the other neighbor without changing them. These messages will be used to set the column ID/ frequency channel of columns.

- **Intermediate Column Head**:

  When an intermediate column head receives a column ID/frequency message, it sets its column ID, and then forwards the message with increased column ID and increased frequency channel to the other node in the core network. After a certain delay it sends a new message that holds the frequency channel of the column as well as the first node ID to sensor nodes in its column. And then switches its operating frequency to the value received.

- **Edge Column Head**:

  An edge column head operates exactly as an intermediate column head but without forwarding the message to nodes in the core network.

- **Intermediate Sensor Nodes**:

  Intermediate sensor nodes wait until the node ID/frequency message is received. They set their node ID, and read the frequency channel from the message. Then, they increase the node ID count and forward the message to the other sensor node in the column. After a certain delay, they switch their operating frequency channel to the new value received from the message.

- **Edge Sensor Nodes**:

  Edge sensor nodes wait until the node ID/frequency message is received. Then, they set their node ID, and after a certain delay, switch to the new operating frequency channel.

After all nodes have changed their operating frequency to the correct value, data collection processes can be started by the column heads [6, 7]. Nodes in the core network are then free to monitor the network as well as send/receive any control messages from the network operator.

3.2 Simulation Environment

Simulations were performed using ContikiOS-v2.6 [50] and emulated Sky motes [34]. The three proposed algorithms were simulated in Contiki's built-in simulator COOJA using networks with 9 nodes per column. Networks of 1, 2, 4, and 6 columns were studied. We have selected 9 nodes per column in order to be symmetric around the central node and with a manageable column size. The same goes for the number of columns. This setup can be then replicated as the network grows.

Additionally, Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) was used in the simulations as the medium access protocol with acknowledgment messages. IEEE 802.15.4 PHY 2.4 GHz [51] was used as the physical layer technology. The channel model used in the simulations is the Multi-path Ray-tracer Medium (MRM) provided by COOJA with free space distance loss, Additive White Gaussian Noise (AWGN), and low fading. We have chosen this model because in a real life scenario, both the nodes as well as the environment are static. Therefore, there will be low or no shadowing/fading affecting the data transmission, which is a reasonable assumption.

For the Neighbor Identification Algorithm, parameters observed were mainly the number of ping messages that each node sends before making a decision as well as the periodic timer, which is the duration between two consecutive pings. The number of ping messages to be sent tested was 10 and 15 messages/node in order to allow a good resolution of the

RSSI value. Lower values were found to make the network to perform badly. Tests were performed with periodic timer as 1, 2, and 3 sec.

As for the Relative Location Algorithm, tests were performed in order to see if the nodes in the network will select the correct node as the core network head. Because losing the setup message (row or column message) is the main cause of error in the setup, the parameter analyzed was the number of maximum retransmissions of a packet before dropping it. Values observed were 3, 5, and 7 retransmissions. In Contiki-OS, the default value is 5. These values were chosen in order to test the limit of the system under different scenarios.

The Frequency allocation algorithm was simulated using the optimum values that were observed from the previous two algorithms.

Simulations were repeated 200 times for each case with each run working until each node identifies its neighbors for the Neighbor Identification Algorithm, until the core network head is selected for the Relative Location Algorithm, and until every node in the network has selected its new operating frequency channel.

3.3 Simulation Results and Analysis

In this section, we show the performance results of the three algorithms in networks obtained from the simulations performed using the COOJA simulator. The parameter observed is the setup error. For the Neighbor Identification Algorithm, the setup is counted as an error if any of the nodes makes a mistake identifying at least one of its neighbors. For the Relative Location Algorithm, a test run is considered correct only if the node selected by the system is the same node defined by the operator, which lies in the center of the network. As for the Frequency Allocation Algorithm, an error is counted if at least one of the nodes in the network has selected a frequency channel different than the one decided by the core network head for the column containing that node. We observed how many of the simulations ended up with the correct decision and which ended up with an error. The three following subsections show the results obtained for each of the algorithms.

3.3.1 Neighbor Identification Algorithm

Table 3.1 shows the error performance of the Neighbor Identification Algorithm. As we can see from the table, for low values of the periodic timer, the error is high. This is due to the congestion that happens from the increased rate of the transmission as well as the increased loss from the collisions of the ping messages.

Table 3.1: Error performance of the Neighbor Identification Algorithm.

Number of Ping Messages	Number of Columns	Periodic Timer		
		1 sec	2 sec	3 sec
10	1	0%	0%	0%
	2	2%	0%	0%
	4	13%	4%	3%
	6	37%	23%	16%
15	1	0%	0%	0%
	2	0%	0%	0%
	4	1%	1%	0%
	6	23%	4.5%	0.5%

We also can observe that increasing the number of ping messages helps in reducing the error dramatically. Increased number of messages leads to an increased resolution when calculating the RSSI and so, better decisions can be made. Additionally, as the number of columns increases, the error increases. This is also caused by the increased probability of collisions that comes from the increased number of messages being sent from the additional nodes in the system.

We can conclude that using 15 ping messages with 3 sec intervals between them can lead to an almost negligible error, even for a network with 6 columns.

3.3.2 Relative Location Algorithm

The error performance of the Relative Location Algorithm is shown in Table 3.2.

This algorithm starts after the Neighbor Identification Algorithm has ended. And so, we have selected the periodic timer as 3 sec and 15 ping messages in order to achieve the lowest error. As we can see from the results, the setup error increases as the network grows. However, even large networks still have an acceptable value of error. Also, 5

Table 3.2: Error performance of the Relative Location Algorithm.

Number of Columns	Max MAC Retransmissions	Error			
		Total	Cause of Error		
			Neighbor Identification	Column Head Selection	Core Network Head Selection
1	3	1%	0%	1%	N/A
	5	0%	0%	0%	N/A
	7	1%	0%	1%	N/A
2	3	1%	0%	1%	0%
	5	1%	0%	1%	0%
	7	1%	0%	1%	0%
4	3	5%	0%	1%	4%
	5	3%	0%	1%	2%
	7	8%	0%	3%	5%
6	3	18%	1%	9%	8%
	5	17%	1%	10%	6%
	7	23%	1%	14%	8%

MAC retransmissions is found to be the optimal value to be used, even for relatively large networks.

3.3.3 Frequency Allocation Algorithm

The error performance of the Frequency Allocation Algorithm is shown in Table 3.3.

Table 3.3: Error performance of the Frequency Allocation Algorithm.

Number of Columns	Error			
	Total	Cause of Error		
		Neighbor Identification	Relative Location	Frequency Allocation
1	0%	0%	0%	0%
2	1%	0%	1%	0%
4	5%	0%	3%	2%
6	22.5%	1%	17%	4.5%

This algorithm starts after both the previous algorithms have ended. And so, we have selected the periodic timer as 3 sec, 15 ping messages, and 5 MAC retransmissions in order to achieve the lowest total error.

The error increases with the number of columns which is expected. However, since the number of messages needed to make a decision is fewer than in the case of the Relative

Location Algorithm, we can see that the frequency error is less than the location error even for a network with 6 columns.

3.4 Testbed Experiments and Results

Three testbed experiments were performed using Sky motes [34] in order to test the operation of the three algorithms in a real scenario. The three tests were selected with different number of motes and with different environments. Experiments were done 100 times each until each node has made a decision on its frequency channel. And as in the simulations, an error was counted if at least one of the nodes in the network has made the wrong decision in each of the three algorithms outcome. The values selected for the parameters were a 3 sec periodic timer, 15 ping messages and 5 MAC retransmissions in order to achieve the lowest error. The experiments are described in more detail in the next subsections.

3.4.1 Indoor Experiment

In this experiment, motes were placed on a wall in an indoor environment as shown in Figures 3.5 and 3.6.

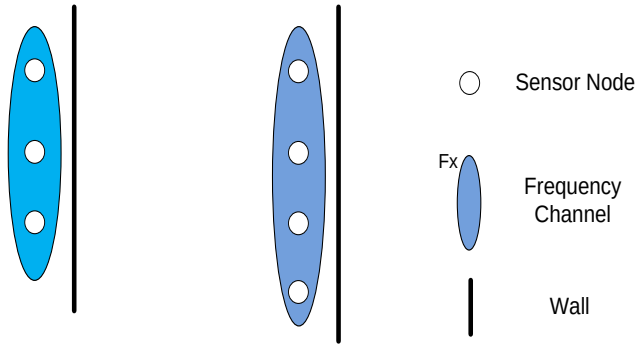


Figure 3.5: Indoor Experiment (a) 3-mote network. (b) 4-mote network.

Distance between nodes was set to 1 meter. Two tests were made, a 3-mote network and a 4-mote network. We have started with an experiment with this simple topology in order to confirm the operation of the Neighbor Identification Algorithm on a small scale.

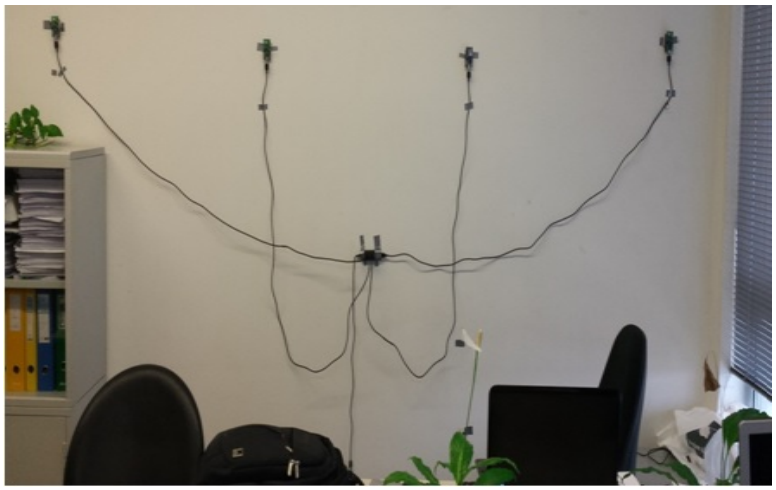


Figure 3.6: Indoor Experiment 4-mote network.

Results from the two other algorithms are not applicable in this experiment due to the small size of the networks.

Tables 3.4 and 3.5 show the results obtained from the 3-mote and 4-mote networks respectively.

Table 3.4: Median RSSI in dBm per node in a 3-mote indoor network.

Node	Neighbor	Number of received messages	Median RSSI(dBm)
1	2	16	-29
	3	16	-34
2	1	16	-27
	3	16	-28
3	1	16	-27
	2	16	-35

The highlighted cells in the tables show the neighbor(s) that were selected by the algorithm as the closest neighbors. They are the nodes with the maximum RSSI with respect to the node in question. As we can see from the table, the algorithm was able to decide correctly which of the nodes are closest to the node in question. The total error is shown in Table 3.13. Out of the 100 runs of the experiment, the error in the Neighbor Identification Algorithm was found to be 10% in the 3-mote network and 15% in the 4-mote network. This error value is higher than the error obtained from the simulation due to the interference from other devices in the office such as wifi routers which use the same

Table 3.5: Median RSSI in dBm per node in a 4-mote indoor network.

Node	Neighbor	Number of received messages	Median RSSI(dBm)
1	2	16	-27
	3	16	-31
	4	16	-40
2	1	16	-25
	3	16	-26
	4	16	-30
3	1	16	-33
	2	16	-27
	4	16	-28
4	1	16	-42
	2	16	-31
	3	16	-27

frequency as well as reflections from the wall and any moving objects. However, it is an acceptable value.

3.4.2 Panels Experiment

In this experiment, 6 motes were connected to two columns of solar panels while a 7th mote was placed between the columns as shown in Figures 3.7 to 3.10.

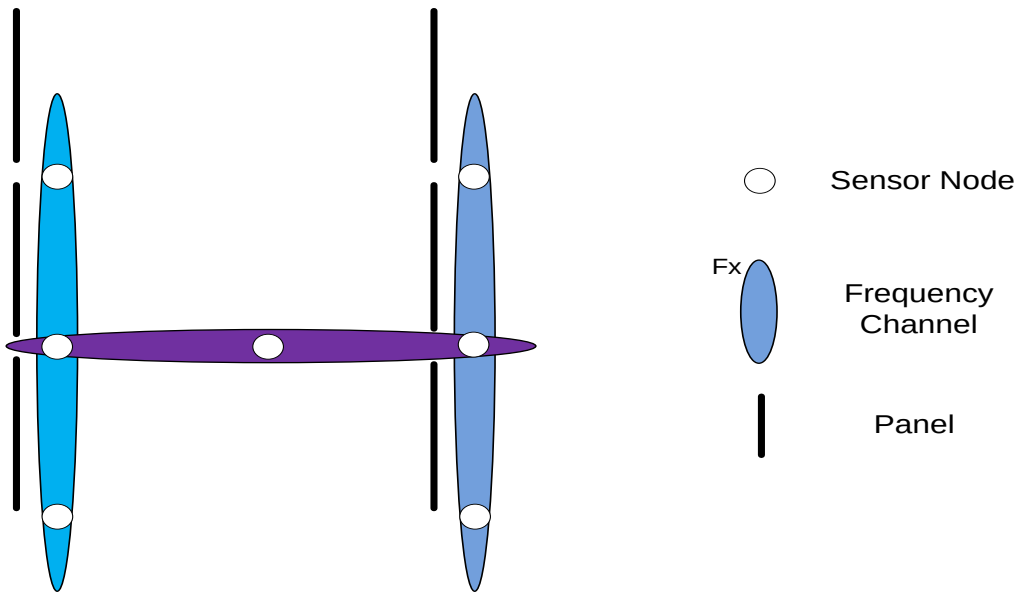


Figure 3.7: Panels Experiment.

Figure 3.8: Panels Experiment (one node).

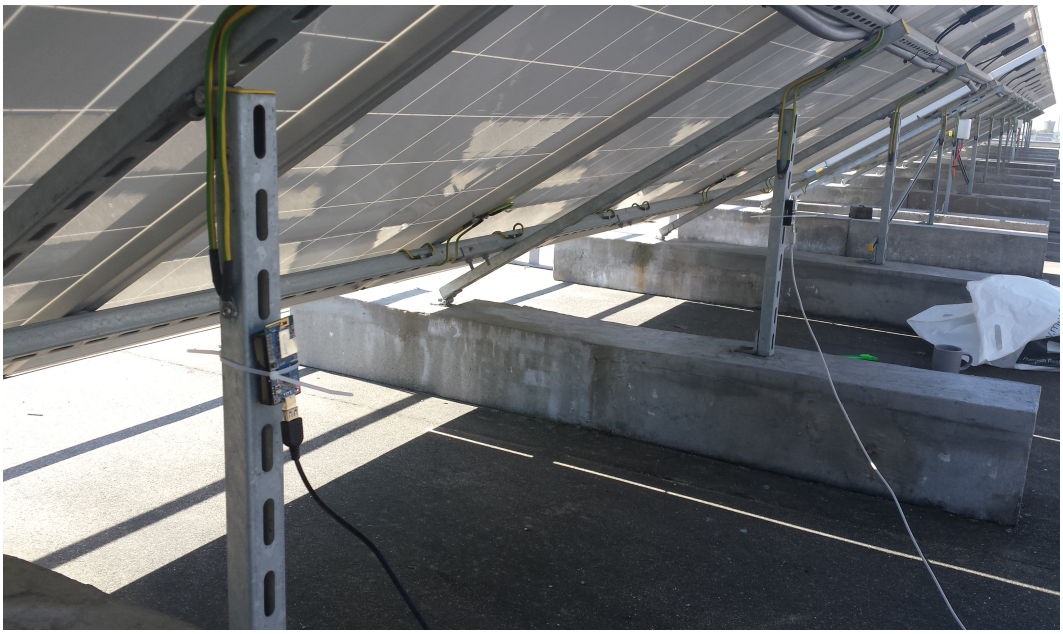


Figure 3.9: Panels Experiment (1st column).

This can be viewed as a subset of the topology shown in Figure 3.1. This experiment was performed on the roof of the INESC TEC building. Distance between motes on the same column was set to 2 meters while the distance between the two columns was set to 4 meters. This experiment was done to test the performance of the algorithm in the presence of an obstacle between the columns. We have placed the motes in the space between two

Figure 3.10: Panels Experiment (2nd column).

consecutive panels in order to reduce the interference coming from the panels.

Tables 3.6, 3.7, and 3.8 show the results obtained from the point of view of a node with one close neighbor, a node with two close neighbors, and a node with three close neighbors, respectively.

Table 3.6: Median RSSI in dBm per neighbor for a node with one close neighbor.

Node	Neighbor	Number of received messages	Median RSSI(dBm)
1	2	16	-27
	3	16	-31
	4	16	-34
	5	1	-34
	6	2	-34
	7	16	-30

As it was stated in the previous experiment, the highlighted cells in the tables show the neighbor(s) that were selected by the Neighbor Identification Algorithm as the closest neighbors. We have only shown results from the point of view of three nodes in order to conserve space. The total error is shown in Table 3.13. The experiment was run 100 times, error in the three algorithms was found to be 17%, 5%, and 2%, respectively. The increase observed in the error value, when comparing with the previous experiment, comes from

Table 3.7: Median RSSI in dBm per neighbor for a node with two close neighbors.

Node	Neighbor	Number of received messages	Median RSSI(dBm)
7	1	16	-31
	2	16	-27
	3	16	-30
	4	2	-34
	5	16	-26
	6	4	-34

Table 3.8: Median RSSI in dBm per neighbor for a node with three close neighbors.

Node	Neighbor	Number of received messages	Median RSSI(dBm)
2	1	16	-27
	3	16	-27
	4	3	-34
	5	12	-34
	6	2	-34
	7	16	-27

the presence of the panels which increased the attenuation of the radio waves and caused some reflections that affected the RSSI calculation and the error in the transmission. However, the algorithms were found to still perform well 76% of the times.

3.4.3 Outdoor Experiment

The outdoor experiment was done with 11 motes. Motes were connected to the top of 1.5m posts placed in an open field. Distance between nodes within the same column was set to 2m while the distance between the two columns was 4m. Additionally, transmission power was increased in order to get a comparable values of results to the previous experiments. The motes created 3 columns with 3 motes each and with two intermediate motes between the columns as shown in Figure 3.11. This also can be viewed as a subset of the topology shown in Figure 3.1. This experiment was done in order to test the performance of the Neighbor Identification Algorithm in the absence of any interference either from the panels or from any moving objects. Figures 3.12 and 3.13, are photos taken of the real testbed.

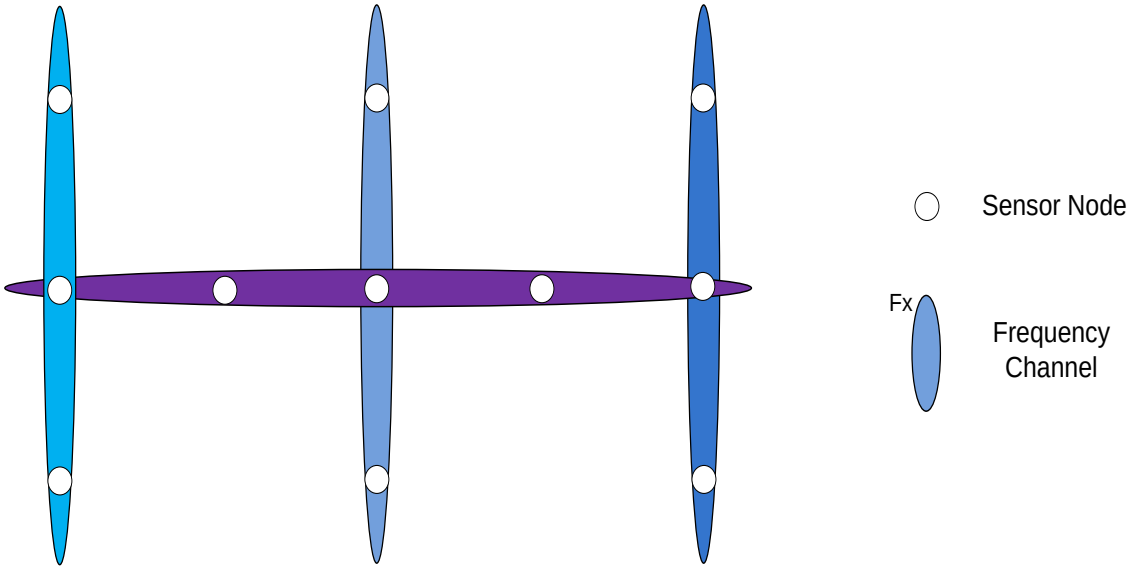


Figure 3.11: Outdoor Experiment.

Figure 3.12: Outdoor Experiment (all nodes).

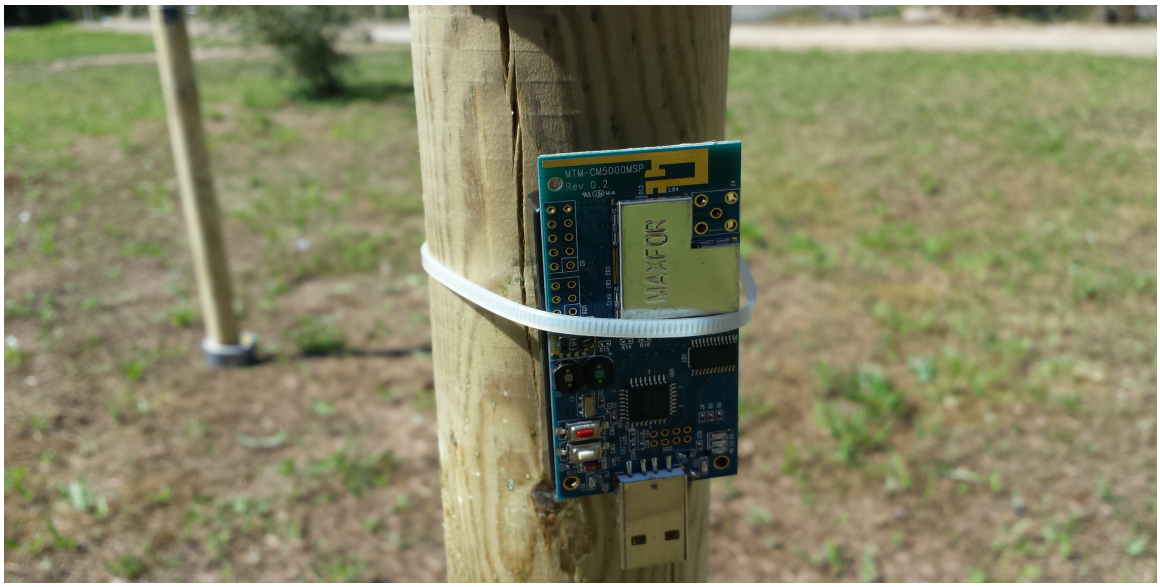


Figure 3.13: Outdoor Experiment (one node).

Tables 3.9, 3.10, 3.11, and 3.12 show the results obtained from the point of view of nodes with 1, 2, 3, and 4 close neighbors respectively.

Table 3.9: Median RSSI in dBm per neighbor for a node with one close neighbor.

Node	Neighbor	Number of received messages	Median RSSI(dBm)
1	2	16	-26
	3	16	-32
	4	16	-31
	5	16	-33
	6	9	-34
	7	2	-34
	8	2	-34
	9	2	-34
	10	16	-28
	11	2	-34

Table 3.10: Median RSSI in dBm per neighbor for a node with two close neighbors.

Node	Neighbor	Number of received messages	Median RSSI(dBm)
11	1	2	-34
	2	2	-34
	3	1	-34
	4	16	-29
	5	16	-26
	6	16	-29
	7	16	-29
	8	16	-25
	9	16	-29
	10	16	-31

Table 3.11: Median RSSI in dBm per neighbor for a node with three close neighbors.

Node	Neighbor	Number of received messages	Median RSSI(dBm)
2	1	16	-25
	3	16	-26
	4	15	-32
	5	16	-31
	6	16	-33
	7	1	-34
	8	2	-34
	9	1	-34
	10	16	-25
	11	4	-34

Table 3.12: Median RSSI in dBm per neighbor for a node with four close neighbors.

Node	Neighbor	Number of received messages	Median RSSI(dBm)
5	1	16	-32
	2	16	-31
	3	15	-32
	4	16	-26
	6	16	-26
	7	16	-33
	8	16	-31
	9	16	-33
	10	16	-25
	11	16	-26

From the 100 experiments, the error in the algorithms' performance was found to be 8%, 2%, and 1%, respectively. These values are small because of the reduced interference from the lack of panels or moving objects in the open field. The total error is shown in Table 3.13.

Table 3.13: Error performance of the algorithms in the testbed experiments.

Testbed	Error			
	Total	Cause of Error		
		Neighbor Identification	Relative Location	Frequency Allocation
3-mote network "indoor"	10%	10%	N/A	N/A
4-mote network "indoor"	15%	15%	N/A	N/A
Panels	24%	17%	5%	2%
Outdoor	11%	8%	2%	1%

3.5 Conclusions

In this Chapter, we have proposed and implemented three algorithms that when used consecutively, allow nodes in a dense wireless sensor network to identify their neighbors, their relative location and select an operating frequency channel that is different from the other nodes within different columns in the network with no external interference from the human operator. The algorithms were implemented and tested using the Contiki-OS and its built-in simulator COOJA. Networks of 1, 2, 4, and 6 columns, each having 9 nodes, were tested. Additionally, three testbed experiments were performed in different environments in order to test the performance of the algorithms in a real scenario. The study was done in order to measure the error percentage of the algorithms.

We were able to conclude that the algorithms allow nodes to configure themselves with no interference from the operator of the network. Results showed that the error in performance decreases as we increase the number of RSSI values used for decision making. Additionally, the number of nodes in the network affects the setup error greatly. However, the value of the error is still acceptable even for high number of simulated columns.

Chapter 4

Data Collection

Usually in WSNs, low data rate is employed. However, other challenging issues related to the reliability of the communication links and to the efficient use of batteries appear in this type of network [3]. Also, because of the "many-to-one" feature of the data collection applications in WSNs, wireless interferences and collisions, and the huge sizes of these networks, the scheduling of data transmissions becomes a challenging problem which needs to be carefully addressed. Another aspect that needs consideration is the data collection technique, i.e., the way sensors send their data to the sink. Each node can send its data to the sink as soon as it is ready, or nodes can wait for a request from the sink before sending their data. As the size of the network grows, the selected data collecting technique will have a huge impact on the performance of the network as a whole.

The work presented in this chapter studies how the data collection technique used to gather information in a well defined scenario affects the performance of a WSN. As stated in the introduction and in the previous chapter, the communications scenario is provided by the SELF-PVP project [49]. This project aims to increase the efficiency of a photo voltaic (PV) power plant. It assumes that 200,000 solar panels are deployed in a matrix like deployment spread over a large area (250 hectares).

After the algorithms described in the previous chapter finish their work, we end up with a group of columns, each operating in a different frequency channel. The work done in this chapter focuses on how the sensor data is collected from each of the sensors and sent to a central node within each column.

In this chapter, we evaluate the performance of an IEEE 802.15.4 wireless sensor network employing three different data collecting techniques within a column where the sink node is located in the center of the column. This client nodes and their sink will be referred to as the network from now on. In Technique 1, client nodes send their data to the sink as soon as the data is ready. In Technique 2, the sink sends a broadcast poll message to its neighbors, which is propagated through the network triggering each client node to send its data to the sink. Technique 3 is similar to Technique 2 except that the poll message is sent to half of the network first and then the sink waits to receive the data from this half before sending the poll message to the other half.

Additionally, we evaluate the performance of Technique 3 with data aggregation in a multi-hop network. Since the information coming from the nodes is similar and headed to the same destination, it is reasonable to assume that data aggregation will improve the performance of the network. The performance is then compared against the performance of a one hop network with no data aggregation. Also, the functionality of a sink command message was enabled. This gives the sink node the ability to communicate with the client nodes after each cycle of data collection in order to send different commands that might be used by the clients.

The study considers different networks, each is made of a column with a different number of nodes and with different values of the offered load, estimating for each network size and offered load, the network throughput, the packet loss and the end-to-end packet delay. The evaluation of the performance of the three techniques was carried out using the Contiki COOJA simulator [26], available at [50].

The rest of the chapter is organized as follows. Section 4.1 presents the topology and the communications architecture proposed for the WSN, as well as the different techniques for data collecting and communication through the WSN. The simulation and tesbed environments are described in Section 4.2. Results are evaluated in Section 4.3. Finally, conclusions are listed in Section 4.4.

4.1 Proposed Architecture and Data Collection Techniques

As stated in the thesis introduction, the objective of this work is to define a communications solution for a solar power plant to allow it to operate at an optimum power delivery point. Each solar panel will have a sensor/actuator made of small and low power wireless sensor node that will sense local variables, communicate these values with other sensors in order to reach the optimum power delivery point of the whole plant. The proposed WSN nodes topology and communications architecture is shown in Figure 4.1.

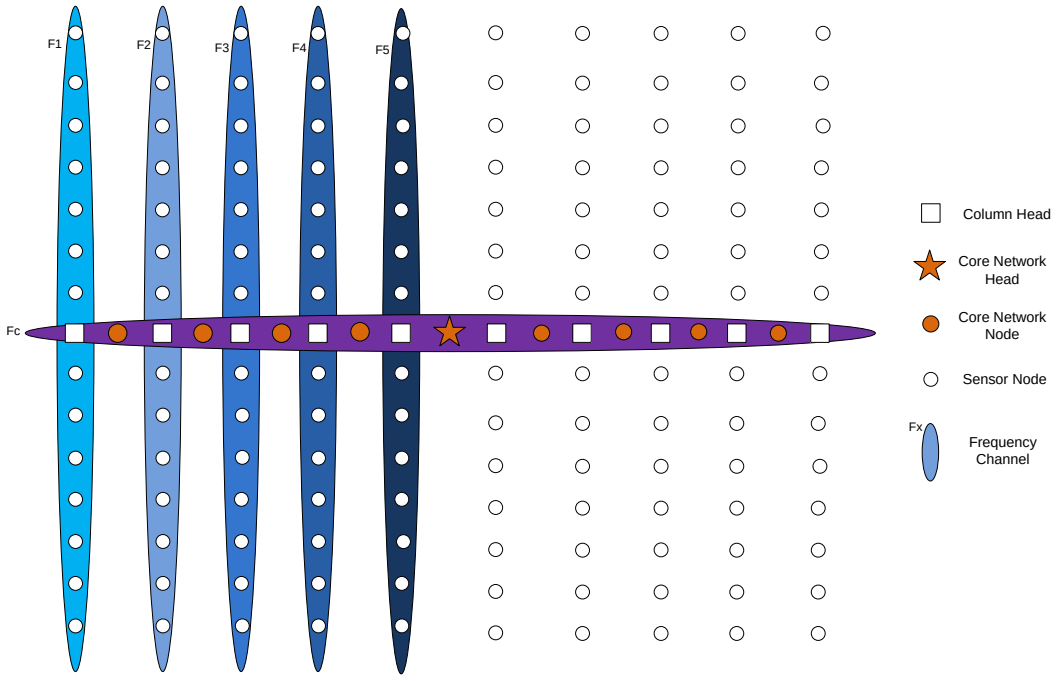
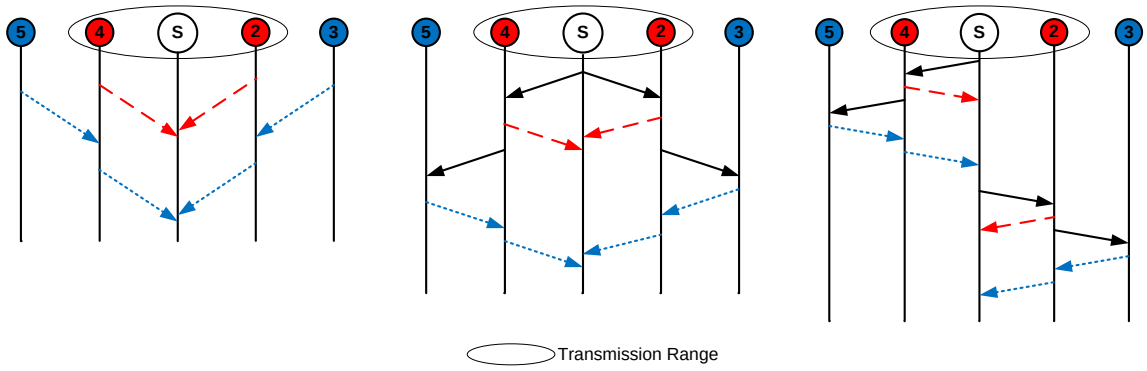


Figure 4.1: Network topology.

The work done in this chapter starts after the self-organization algorithms, discussed in the previous chapter have concluded. We assume that these algorithms have managed to configure the network correctly and were able to divide it into a group of columns where each column is considered a smaller network operating in a different frequency channel than the rest of the columns. Within each column, a data collection technique will have to run in order to collect the information from the sensors and send it to a central node for processing. Throughout this chapter, three techniques will be analyzed for data collection and communication between the nodes.

Figure 4.2 shows an example of how these techniques work in a network with a sink and 4 client nodes.

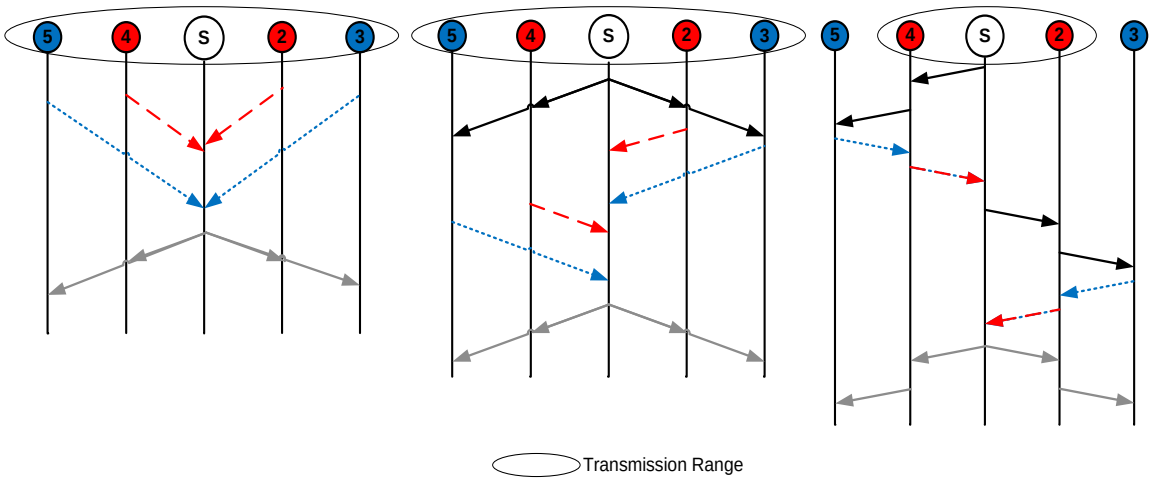


(a) Technique 1. (b) Technique 2. (c) Technique 3.

Figure 4.2: Proposed data collection and communication techniques.

In Technique 1, client nodes send data to the sink whenever they are ready which is the basic way of communications in WSNs. In Technique 2, the sink sends a broadcast poll message to its neighboring nodes and each node replies with its data as soon as it receives this poll before forwarding it to its other neighbors, thus propagating the poll message across the network. This is done in order to reduce collisions between the messages. Finally, in Technique 3, the procedure is similar to Technique 2, excepts that the sink first sends the poll to one of its neighbors and waits until it receives the information from all the nodes on this side of the network before sending the poll to the other neighbor in order to receive data from the rest of the nodes on the other side of the network. The messages shown in Figure 4.2 are only the application layer packets; we rely on the MAC layer to perform the acknowledgments and retransmissions of messages when collisions occur.

Additionally, we have tested the effect of data aggregation on Technique 3, and compare it with the performance of a one hop network employing Techniques 1 or 2 while also enabling the sink command message. This command message enables the sink to communicate with the nodes after each cycle of data collection, to send a command that can be used in the next cycle. These techniques after the change will be referred to as the modified techniques and are shown in Figure 4.3.



(a) Modified Technique 1. (b) Modified Technique 2. (c) Modified Technique 3.

Figure 4.3: Modified data collection and communication techniques.

In modified Technique 1, the nodes are direct neighbors of the sink, and so, they send their data directly to it. After the data is collected from all the nodes, the sink sends a broadcast command message to all the nodes. In modified Technique 2 the broadcast poll is sent to all the neighbors, the nodes then take turn sending their data to the sink. And finally, the sink sends the broadcast command message to all of the nodes. Technique 3 has been modified to include data aggregation. When a node receives the poll message, it forwards it to its neighbor and waits until it receives data from this neighbor. After that, it adds its data to the received message in order to create an aggregated message, which is then forwarded in the direction of the sink. The sink then repeats this procedure to the other side of the network, and then sends the broadcast command message to its neighbors, which is then forwarded to the rest of the nodes.

Modified Techniques 1 and 2 were tested in a one hop setup as this is the setup that achieves the highest throughput of data, since every message can reach the sink in one hop. The motivation behind this is to prove that modified Technique 3, while operating in a multi-hop setup with low transmission power and so low interference, can perform close to the other two modified techniques.

4.2 Simulation and Testbed Environments

Simulations have been done using Contiki2.5 [50] and Sky motes [34]. The three data collection techniques were simulated in Contiki's built-in COOJA simulator using different numbers of nodes in the network (4, 8, 12 and 16) and for different values of the offered load (1, 2 and 4 packet/s), where the offered load is the data rate that each client node is offering to the network and total offered load is the sum of the offered loads from all the client nodes in the network. Inter packet arrival times were made sure to be exponentially distributed in order to simulate a real life scenario.

The three modified techniques were tested using both simulation and a testbed using a network with a sink and 8 client nodes. We have selected 8 clients for the network to be symmetric around the sink and with a manageable network size. This setup can be replicated as the network grows. The performance of the techniques was measured for different values of the offered load. These values were chosen in order to test the limit of the system.

Simulations were repeated 10 times for each case with each run working until the sink receives a total of 10000 packets in the case of the simulations and 4000 packets in the case of the testbed experiments. Mean values were calculated with a 90% confidence interval.

Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) was used in the simulations as the medium access protocol with acknowledgment messages. IEEE 802.15.4 PHY 2.4 GHz [51] was used as the physical layer technology. The channel model used in the simulations is the Unit Disk Graph Medium (UDGM) provided by COOJA with free space distance loss and no fading. We have chosen this model, because in a real life scenario, both the nodes as well as the environment are static. And so, there will be low or no shadowing/fading to affect the data transmission, which is a reasonable assumption. In modified Technique 3, nodes were placed apart so that each node has always two neighbors falling within its transmission and interference ranges. This was done easily in the simulator by controlling the transmission and interference ranges of the nodes in simulation environments. However, in the testbed, small modification had

to be made in the radio driver in order to reduce the receiver sensitivity as well as the transmission powers in order to assure that each node has only two neighbors falling within its transmission range.

4.3 Results and Analysis

Different parameters were observed throughout the experiments performed in this chapter. Below, we list these parameters and explain how they were calculated.

Throughput is calculated per node as the number of received packets at the sink from that a node, divided by the experiment time. Then this value is averaged over the total number of nodes to get the average throughput.

Total delay is the time needed to collect the data from all the nodes in the network.

The total packet loss is the total number of transmitted packets from all the nodes minus the total number of received packets at the sink divided by the total number of transmitted packets from all the nodes.

Packet loss per node is calculated as the number of transmitted packets from this node minus the number of received packets at the sink from the same node divided by the number of transmitted packets from the same node.

To be clear, we are counting only the IPv6 packets that arrive correctly at the destination. The retransmitted layer two frames are not counted when calculating the packet loss. Additionally, the load is defined here as the data rate that each node is offering to the network. Next we show the results gathered from these experiments for the different techniques proposed.

4.3.1 Link Analysis

Figure 4.4 shows the throughput of a network with one client for different values of offered load. The client sends data to the sink with a constant rate. This rate is changed gradually in order to test the capacity of the system. From the figure we can see that the throughput increases linearly with the offered load until a point with 47.76 packet/s at a value of offered load of 50 packet/s after which it saturates. This can give us an idea about

the limitation of the system, which can help with the rest of simulations by establishing a base line of the maximum throughput a single node can achieve in this scenario.

These results coincide with the results we got in [6], in which the one hop delay was 20 ms per packet per node, which means that a node can send up to 50 packet/s. Which is close to the link analysis performed.

We are assuming that hop delay comes mainly from the packet transmission delay, medium access delay, and processing delay. The propagation delay for such small distances between the nodes is very low and so can be neglected.

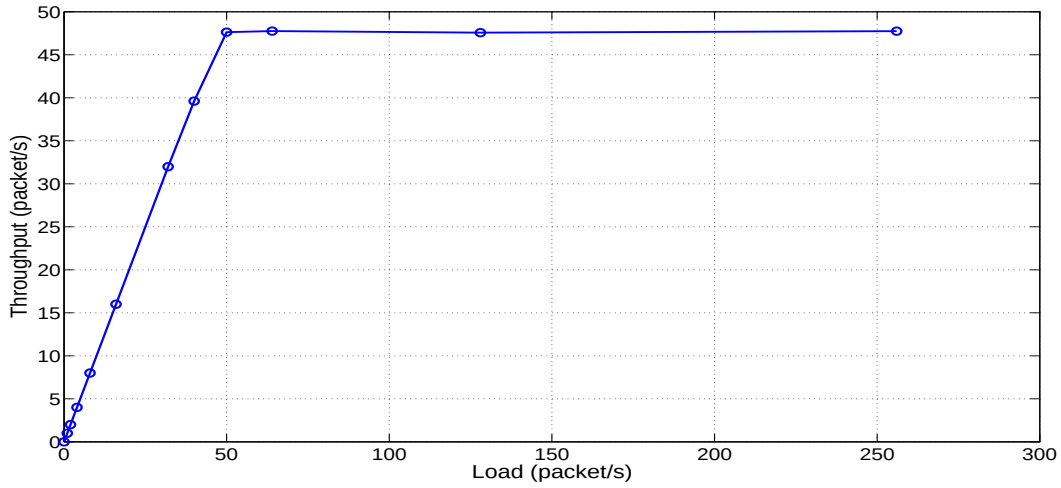


Figure 4.4: Throughput vs. offered load for a system with one client node.

4.3.2 Performance of the Techniques

Figure 4.5 shows the average throughput in packet/s for an offered load of 1 packet/s/node. It is plotted against the number of simulated nodes in each simulation for the three techniques and the offered load. It is clear from the figure that, as the number of nodes increases, the throughput decreases due to the error coming from collisions and from retransmissions. However, Technique 3 operates the closest to the offered load limit in all the cases, while Technique 2 performs worst.

Figure 4.6 shows the average throughput in packet/s for an offered load of 2 packet/s/node. As we can see, curves follow the same trend as for the 1 packet/s/node offered load. However, the throughput decreases much faster as the number of nodes increases. Technique 3

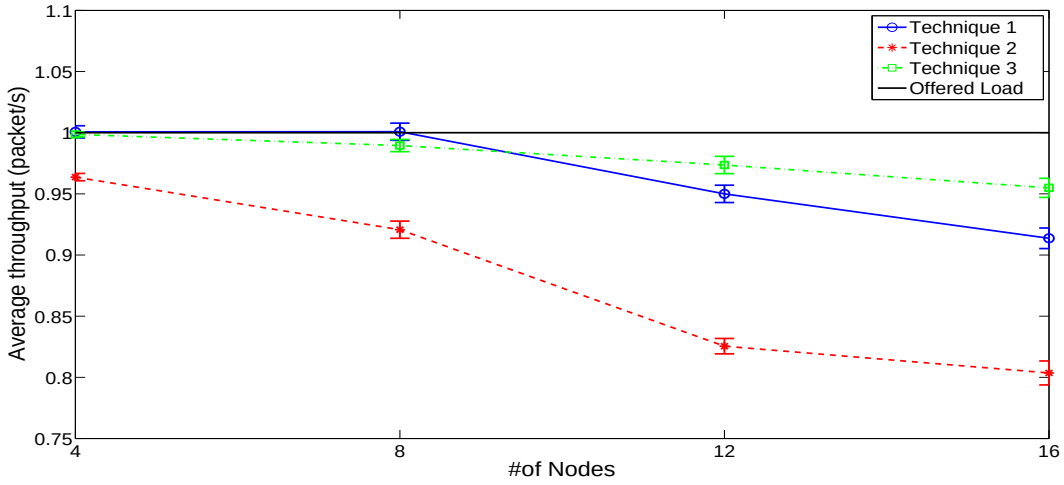


Figure 4.5: Average throughput for an offered load of 1 packet/s/node.

still performs best.

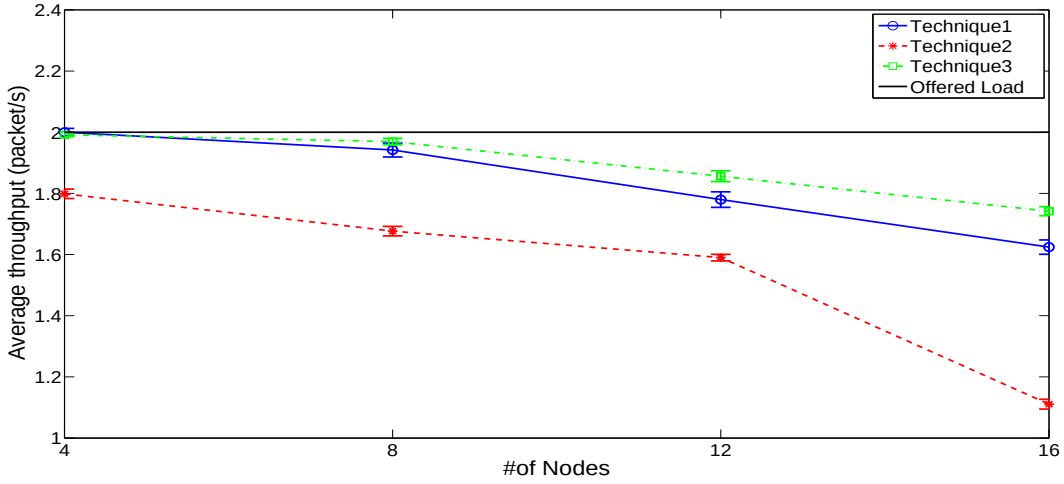


Figure 4.6: Average throughput for an offered load of 2 packet/s/node.

Figure 4.7 shows the average throughput in packets/s for an offered load of 4 packet/s/node. Throughput decreases even faster with the increase of the number of nodes in the network. In this case, Technique 1 and 3 perform very close to each other. This is mainly due to the increased number of collisions that comes from the increased number of transmitted packets. Technique 2 still performs worst. Comparing this figure with the two previous figures, we can see that as the offered load increases the decrease in the throughput becomes faster in all the techniques.

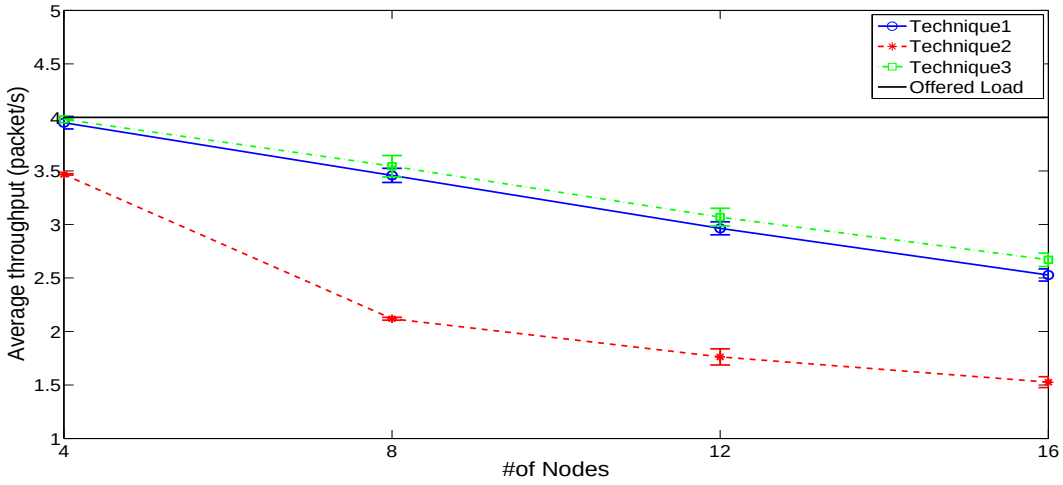


Figure 4.7: Average throughput for an offered load of 4 packet/s/node.

Figure 4.8 shows the total packet loss as a percentage of the total number of packets generated by the network, for an offered load of 1 packet/s/node. We can see that packet loss increases as the number of nodes in the network increases, and with the increase of the offered load. However, Technique 3 performs much better than the other two techniques due to the fact that it tries to avoid collisions by design. Technique 2 suffers more losses because the two nodes neighboring the sink have a high probability of collisions as they send their reply to the sink almost at the same time.

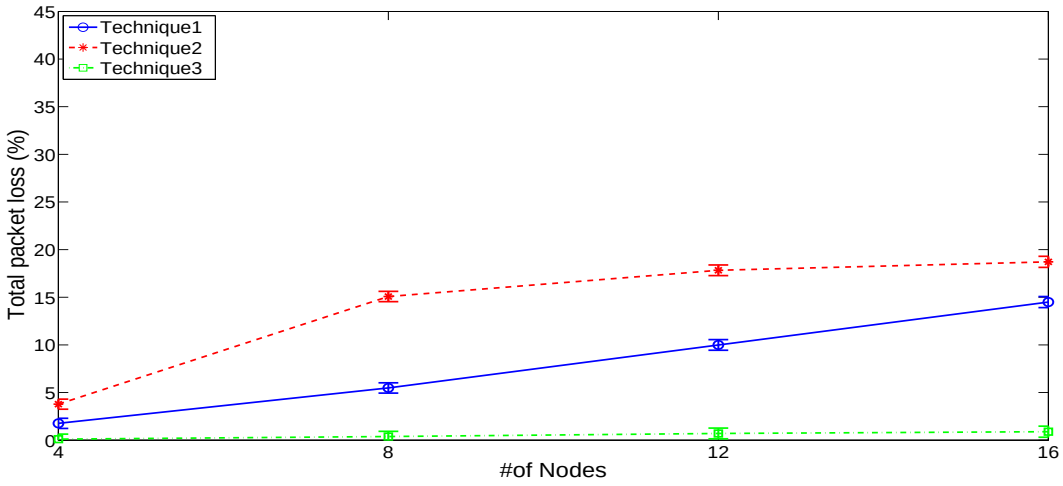


Figure 4.8: Total packet loss for an offered load of 1 packet/s/node.

Figure 4.9 shows the total packet loss for an offered load of 2 packet/s/node. As the

offered load increases, the loss increases for all the techniques. While techniques 1 and 2 act similar to the previous case, Technique 3 still almost has no loss for low number of nodes, but as this number increases, the loss increases greatly.

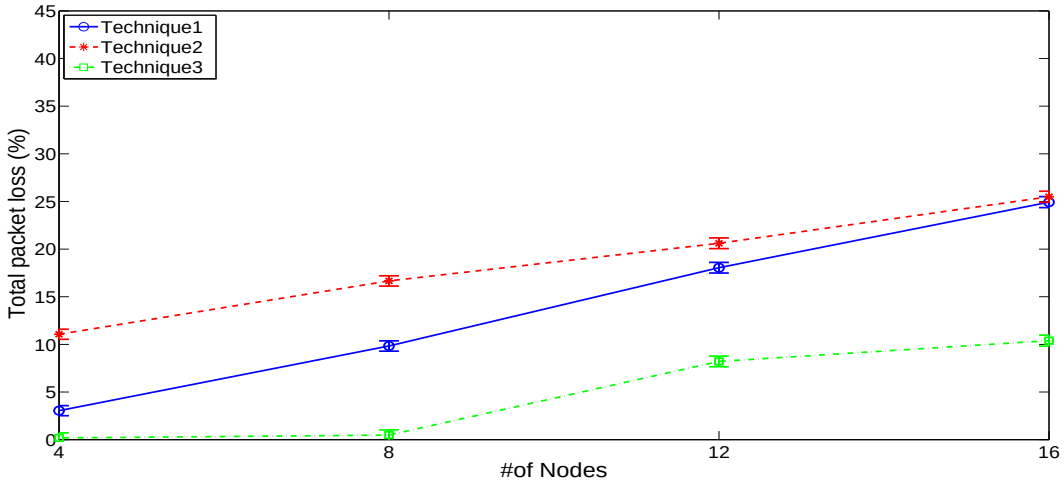


Figure 4.9: Total packet loss for an offered load of 2 packet/s/node.

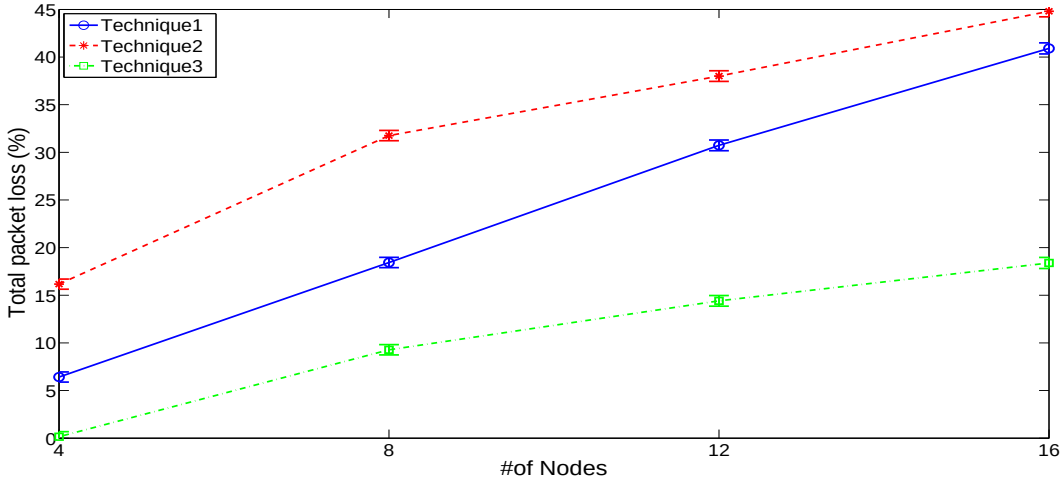


Figure 4.10: Total packet loss for an offered load of 4 packet/s/node.

Figure 4.10 shows the total packet loss for an offered load of 4 packet/s/node. We can clearly see the gain Technique 3 has over the other techniques in terms of packet loss. While techniques 1 and 2 have loss up to 40%, Technique 3 has only 15% losses.

Figures 4.11, 4.12, and 4.13 show the total delay in milliseconds for offered loads of 1, 2, and 4 packet/s/node, respectively.

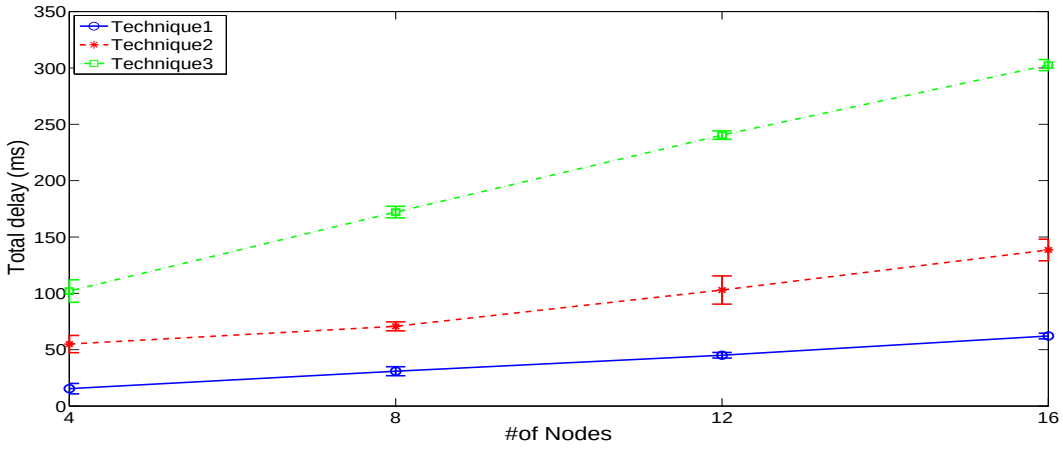


Figure 4.11: Total delay for an offered load of 1 packet/s/node.

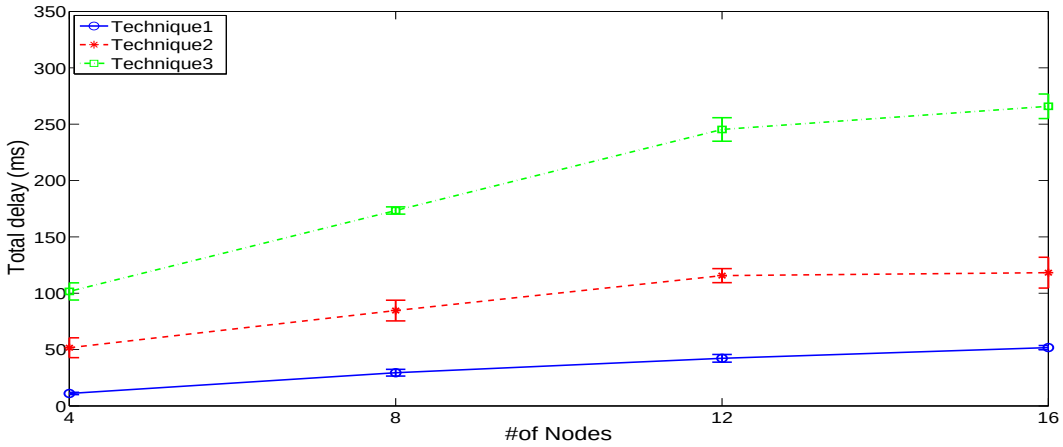


Figure 4.12: Total delay for an offered load of 2 packet/s/node.

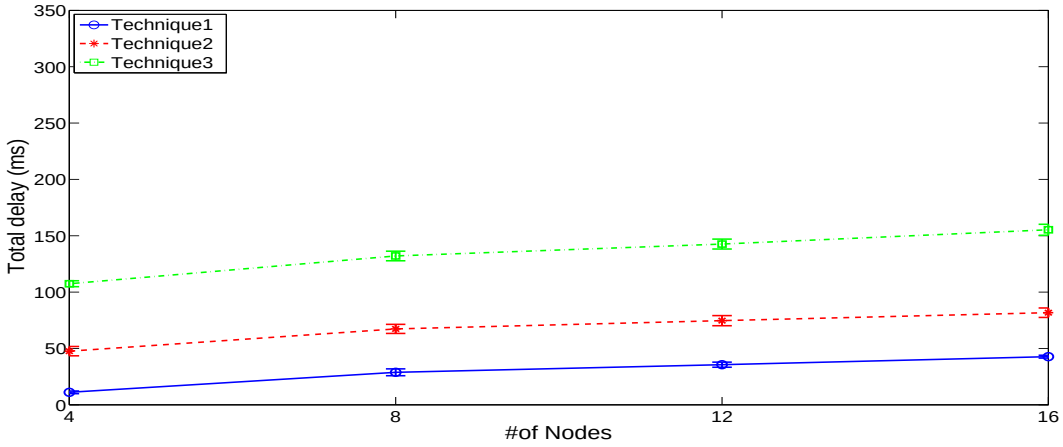


Figure 4.13: Total delay for an offered load of 4 packet/s/node.

We can see that the total delay is not affected greatly by the increase in offered load for the values considered. It depends more on the number of nodes in the network. Additionally, Technique 1 clearly has the smallest delay because nodes start sending their data without waiting for a message for the sink, while the other two techniques have more delay, because they are based on a polling mechanism.

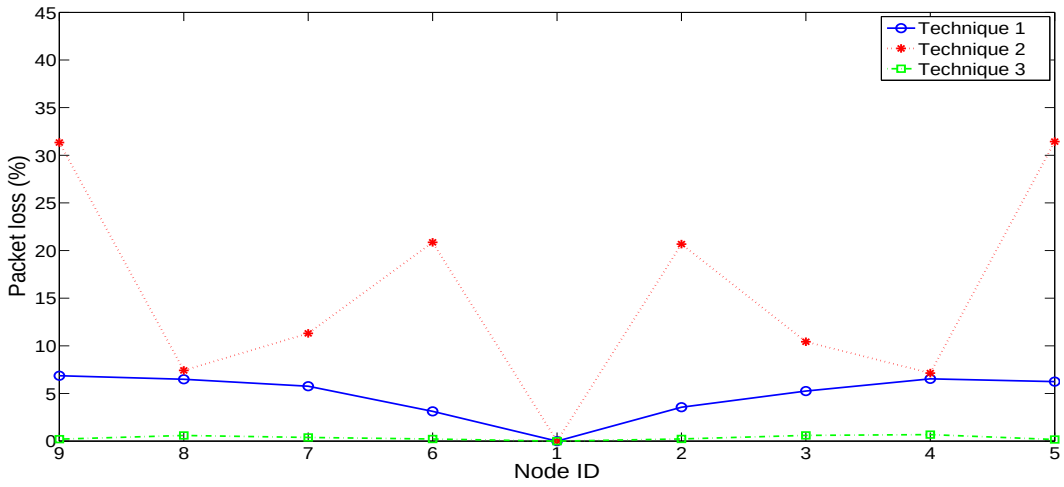
Figures 4.14 (a), (b), and (c), show the packet loss per node against the node ID for a network with 8 client nodes with the sink being node 1, for offered loads of 1, 2, and 4 packet/s/node, respectively. Losses increase as we move away from the sink when using Technique 1. In Technique 2, nodes neighboring the sink suffers high losses because they are hidden from each other and so their packets collide with higher probability. Technique 3 performs best because it tries to avoid collisions by design. However, as the load increase, further nodes start to suffer high losses.

From what we can see from these results, Technique 3 performs best in terms of throughput and packet losses. However, due to the polling mechanism, data collection takes much longer than Technique 1.

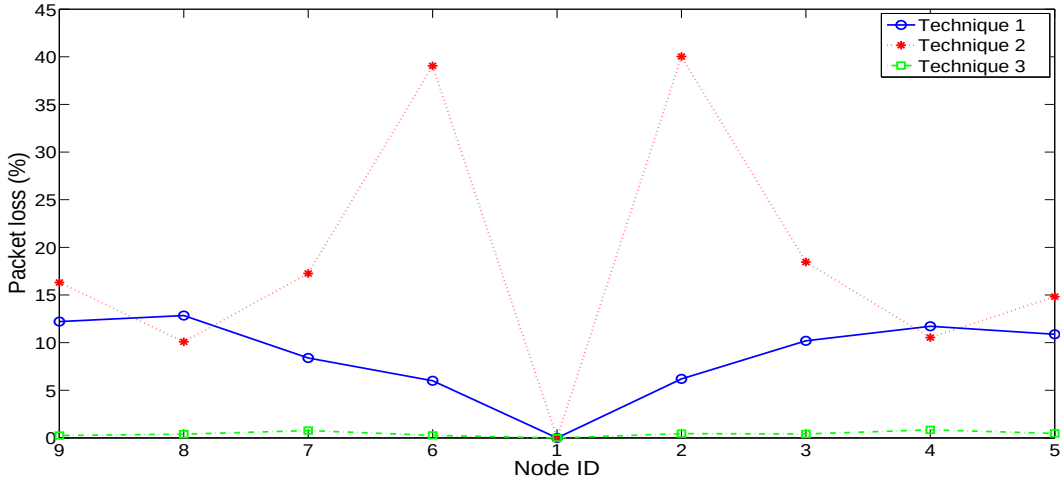
Technique 2 performs worst overall due to the high error from collisions between the two nodes neighboring the sink. They are considered “hidden” from each other and so their packets collide with high probability. Thus, this technique will be abandoned in further testing.

For Technique 3, a conclusion we got during the design is that the offered load and the number of nodes are connected. In order to avoid collisions, the time between polls has to be larger than the time it takes to collect the data. If this is not realized, loss will increase for the further nodes, which will reduce the total throughput of the system.

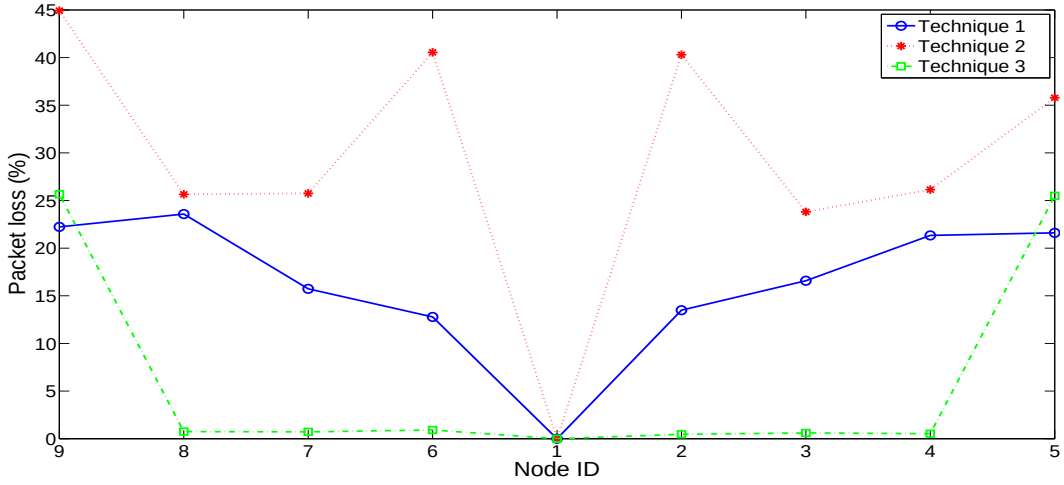
As the load increases, performance degrades for all the techniques. However, Technique 3 still outperforms the other techniques in terms of throughput and packet loss.



(a) Packet loss per node for an offered load of 1 packet/s/node.



(b) Packet loss per node for an offered load of 2 packet/s/node.



(c) Packet loss per node for an offered load of 4 packet/s/node.

Figure 4.14: Node packet loss of a network with 8 client nodes for different values of offered load.

4.3.3 Performance of the Modified Techniques with no Sink Command Messages

Figure 4.15 shows the throughput of a network with 8 clients for the three modified techniques for different values of the offered load with no sink command messages. Results show that the throughput of a one-hop network using modified Techniques 1 and 2 saturates at 5.3 and 5.5 packet/s/node, respectively, for a value of offered load of 6 packet/s/node. A multi-hop network using modified Technique 3 with data aggregation, operates close to the other two modified techniques until a value of offered load of 5 packet/s/node after which the performance degrades. This is due to the fact that as the value of the offered load increases, the time between the polls decreases, and so there will come a point when the next poll is sent before all the nodes had time to send their data to the sink. This will lead to congestion which increases the packet loss and causes the throughput to decrease.

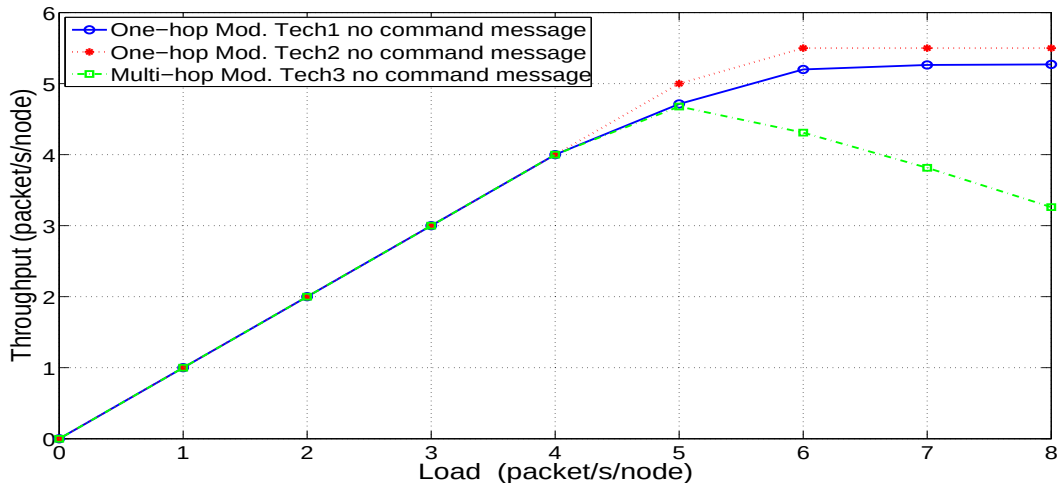


Figure 4.15: Throughput vs. offered load for an 8 clients network for the modified techniques with no sink command messages.

Using the results of the link analysis found in subsection 4.3.1, an approximation of the maximum average throughput of a network with 8 nodes can be roughly estimated by dividing the one node network maximum throughput by 8, which gives us 6.25 packet/s/node. This is close to the results found from the simulations.

4.3.4 Performance of the Modified Techniques with Sink Command Messages enabled

Figure 4.16 shows the throughput of a network with 8 clients for the three modified techniques for different values of the offered load with the sink command messages enabled. We can see that the overall throughput decreases a little when the sink command messages are enabled, which is due to the increased number of messages to be sent in each cycle, which in turns leads to more congestions and packet losses. However, the difference is not that big and we can see that the trend of the results is the same, with modified Techniques 1 and 2 saturating at a value of 6 packet/s/node offered load and with 5.2 and 5.3 packet/s/node, respectively. And with modified Technique 3 performing close until 5 packet/s/node of offered load.

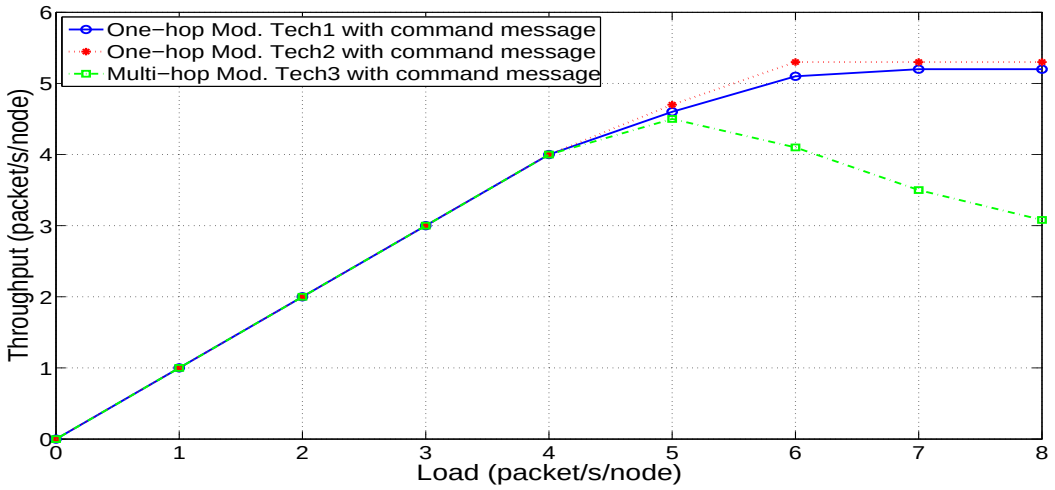


Figure 4.16: Throughput vs. offered load for an 8 clients network for the modified techniques with sink command messages enabled.

Figure 4.17 shows the sink command message throughput of a network with 8 clients for the three modified techniques for different values of the offered load with the sink command messages enabled. The sink command messages throughput is calculated as the mean number of sink command messages received by the nodes divided by the simulation time. The sink command message contains information from all nodes in the network which means that its size is equal to the size of the full aggregated packet in the case of the modified Technique 3. We can see from the figure that the trend is the same as the

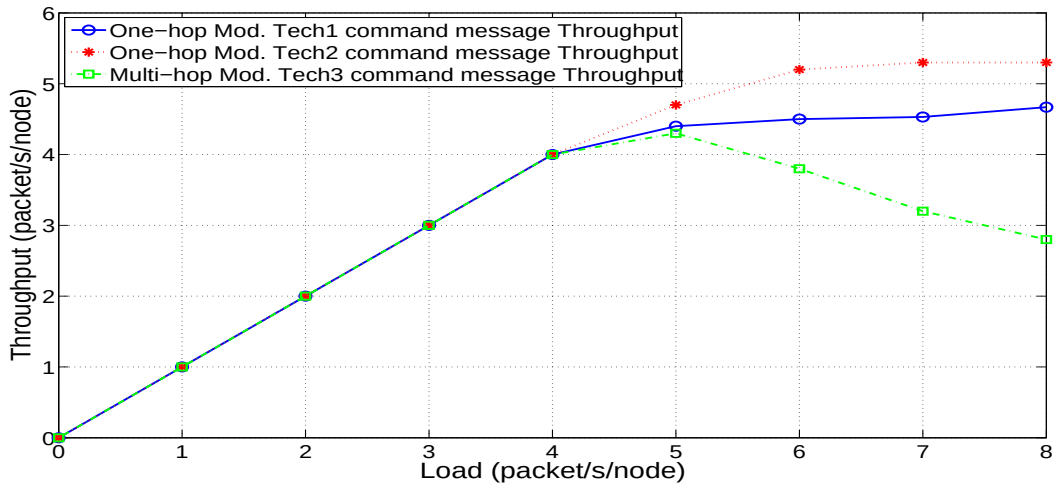


Figure 4.17: Command message throughput vs. offered load for an 8 clients network for the modified techniques with sink command messages enabled.

previous figures.

4.3.5 Packet Loss Per Node of the Techniques with no Sink Command Messages

Figures 4.18, 4.19, and 4.20 show the packet loss per node against the node ID for a network with 8 clients with the sink being node 1 for the three modified techniques for different values of offered load with no sink command messages. Values of offered load are 6, 7, and 8 packets/s/node. The reason for showing only these values is because the loss is almost zero for the lower values of offered load. We can see from the figures that while modified Technique 3 performs worse than the other techniques for these values of offered load, the loss is almost constant for all the nodes in the system. This means that there is a degree of fairness in the network as all the nodes transmit the same amount of data. This is different for the other techniques where nodes closer to the sink have low values of packet loss compared with nodes that are far from the sink.

We can also observe from the figures that as the offered load increases, the loss difference between the two branches of the network increases as well, with the branch to be polled second having higher losses. This is due to the fact that as the load increases, the time taken to poll the nodes decreases. And so, nodes on the second branch will have less time to reply which leads to higher losses.

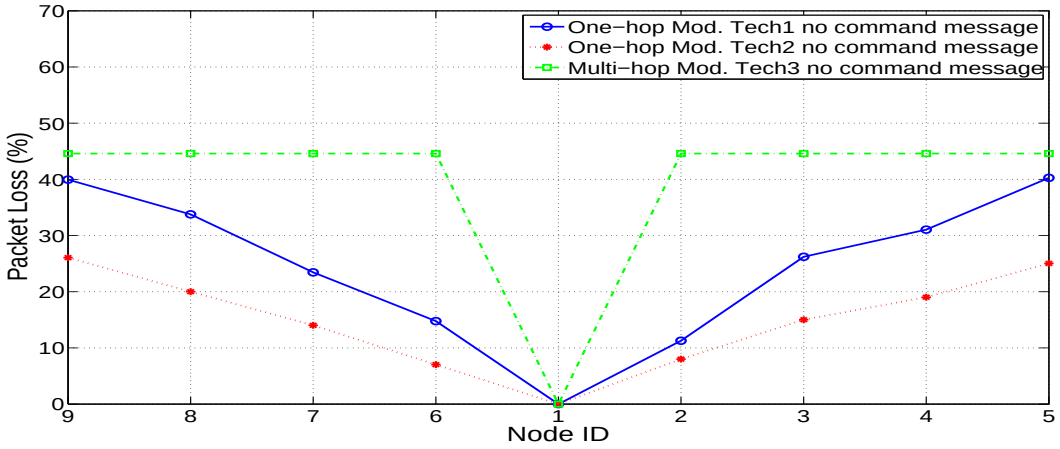


Figure 4.18: Packet loss per node for an offered load of 6 packet/s/node.

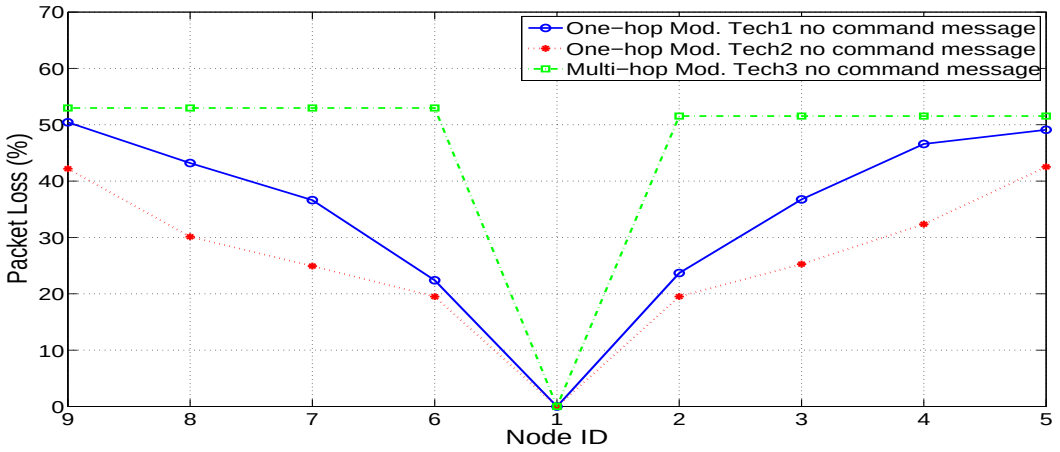


Figure 4.19: Packet loss per node for an offered load of 7 packet/s/node.

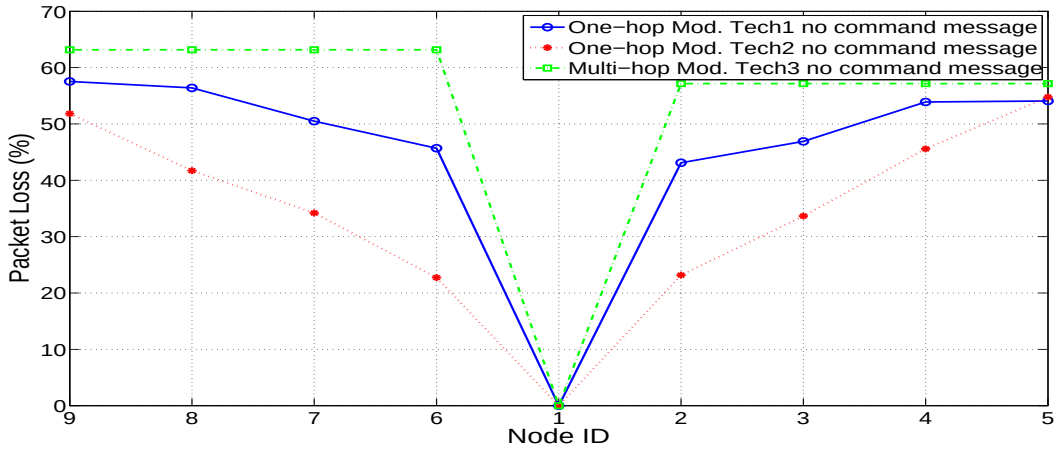


Figure 4.20: Packet loss per node for an offered load of 8 packet/s/node.

From what we can see from these results, Technique 3 with data aggregation in a multi-hop network, performs close to the other techniques in a one-hop network and no aggregation up to a value of the offered load after which its performance degrades severely due to the congestion.

We have selected to use a multi-hop network in order to reduce transmission power as well as interference levels among the different nodes, and with Technique 3 performing close to the others in a multi-hop network, this shows that it is the best choice for the requirements of the network.

However, we must take into consideration that this performance degrades as the offered load increases beyond a certain point. This is due to the congestion that happens in the network which in turns adds to the packet loss.

4.3.6 Testbed Experiments

In order to validate the results obtained from the simulation of modified Technique 3, a testbed was created. The testbed consisted of 8 client nodes and a sink placed in a line with the sink in the middle following the same topology used in the simulations. The transmission power was reduced and nodes were placed relatively far apart in order to ensure that each node has only two direct neighbors in order to create a multi-hop network. Additionally, receiver sensitivity was reduced in order to ignore any packets coming from nodes who were not direct neighbors.

Figure 4.21 shows the throughput of both Technique 3 and modified Technique 3 against the offered load with command message disabled. The figure also shows results obtained from both the simulation and the testbed experiments.

As we can see from the figure, the testbed results confirm the results obtained from the simulation in both the original and modified Technique 3. The testbed results are slightly worse in terms of the throughput due to the interference, which is found in nature from other devices operating in the real world and was not taken into consideration while performing the simulation. Additionally, the figure clearly shows the improvement the aggregation gives to the performance of the modified technique when compared with

the original technique with no aggregation. This improvement comes from the reduced number of messages that are needed to be transmitted with the use of aggregation.

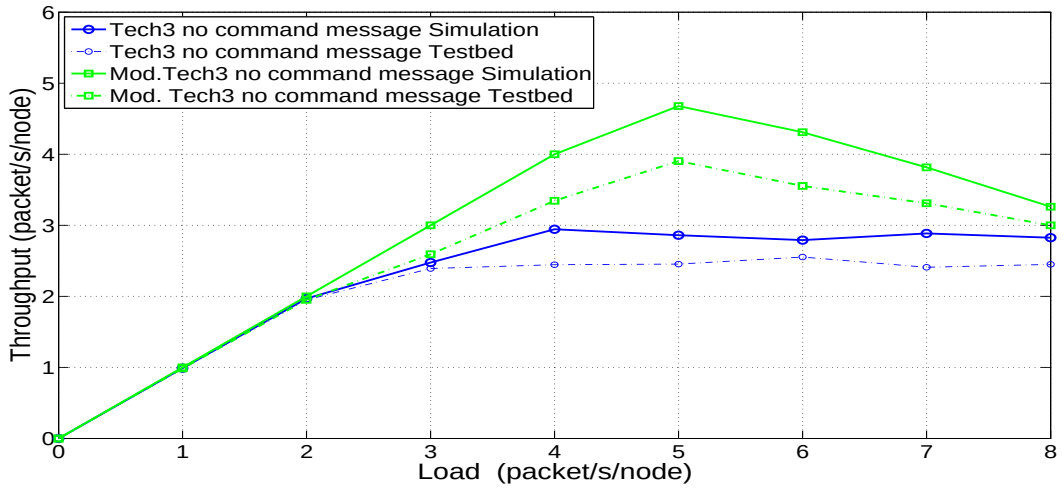


Figure 4.21: Throughput vs. offered load for an 8 clients network for modified and original Technique 3 with no sink command message.

4.4 Conclusions

In this chapter, we studied the performance of an IEEE 802.15.4 strip based wireless sensor network employing three different data collecting techniques using the Contiki COOJA simulator and validating some of the results with testbed experiments using Sky motes. The study was done for different network sizes and different values of the offered load, estimating for each network size and offered load, the network throughput, the packet loss and the end-to-end packet delay.

The evaluation of the three techniques enabled us to conclude that Technique 3 allows a set of nodes in a column to communicate between themselves with low probability of packet loss and high values of average throughput. Additionally, we were able to conclude that Technique 3 with data aggregation achieves the same level of performance up to a considerable value of the offered load. Results showed that as the size of the network grows, Technique 3 performs best in terms of packet loss and throughput. However, this will lead to having a larger overall delay for the data collection process.

As the offered load increases beyond a certain point, the performance of modified Technique 3 degrades due to congestion of the network, which in turns adds to the packet loss. Besides that, enabling the sink command messages also degrades the performance a little due to the added number of messages to be sent. On the other hand, sink command messages might be needed if the sink is required to communicate with the clients, sending different commands, which can be used in the next cycle. Additionally, results obtained from the simulation of both Technique 3 and modified Technique 3 were validated by the use of the testbed. These results have shown that Technique 3 with data aggregation in a multi-hop network performs close to the other two techniques with no aggregation in a one-hop network. We have selected to use a multi-hop network in order to reduce transmission ranges as well as interference levels among the different nodes. And with modified Technique 3 performing close to the others in terms of throughput in a multi-hop network, we were able to show that it is the best choice for the requirements of the network.

Chapter 5

Conclusions

In this concluding chapter, we summarize the content of the thesis and we discuss possible future research directions.

5.1 Overview of the Work

In this PhD thesis, we aimed to create a complete communications solution for a dense wireless sensor network. Throughout the work, we have focused on solving the problem of the self-organization of the sensors in clusters, which also included their auto-configuration and frequency planning. Furthermore, we have worked to enable the efficient broadcast based data collection and communications in such a large network, using simple communications techniques for the transmission and routing of data, while preserving some degree of reliability.

In Chapter 3, we have proposed and implemented three algorithms that, when used consecutively, allow nodes in a dense wireless sensor network to identify their neighbors, their relative location and select a different operating frequency channel than the other nodes within the network, with no external interference from the human operator. The algorithms were implemented and tested using the Contiki-OS and its built-in simulator COOJA. Networks of 1, 2, 4, and 6 columns, each having 9 nodes were tested. Additionally, three testbed experiments were performed in different environments in order to

test the performance of the algorithms. The study was done in order to measure the error percentage of the algorithms.

We were able to conclude that the algorithms allow nodes to configure themselves with no interference from the operator of the network. Results showed that the error in performance decreases as we increase the number of RSSI values used for decision making. Additionally, the number of nodes in the network affects the setup error greatly. However, the value of the error is still acceptable even for high number of simulated columns.

In Chapter 4, we studied the performance of a wireless sensor network in a strip based topology when using one of three different data collecting techniques. This study was performed with the help of the Contiki operating system and its built-in COOJA simulator. Furthermore, some of the results obtained from the simulations were validated with the use of a testbed using Sky motes. Different network sizes and different values of the offered load were tested, estimating for each network size and offered load, the network throughput, the packet loss and the end-to-end packet delay.

We were able to conclude from the results that from the three techniques, Technique 3 was found to allow the set of nodes in a column to communicate with each other with low probability of packet loss and high values of average throughput. Moreover, Technique 3 with data aggregation was shown that it can allow the same level of performance up to a considerable value of the offered load. Results showed that, Technique 3 performs best in terms of packet loss and throughput even as the size of the network increases. However, this will lead to having a larger overall delay for the data collection process. When data aggregation is added to Technique 3, it was shown to perform in a multi-hop network close to the other two techniques with no aggregation in a one-hop network. A multi-hop network was selected in order to reduce transmission ranges as well as interference levels among the different nodes. And with modified Technique 3 performing close to the others in terms of throughput in a multi-hop network, we were able to show that it is the best choice for the requirements of the network.

Another observation that was found is that, as the offered load increases beyond a certain point, the performance of modified Technique 3 degrades. This degradation is due to

congestion of the network which in turns adds to the packet loss. Additionally, enabling the sink command messages also degrades the performance a little due to the added number of messages that are being sent. But still, sink command messages can be useful if the sink is required to communicate with the clients, for example, sending different commands to be used in the next cycle of the data collection. Finally, results obtained from the simulation of both Technique 3 and modified Technique 3 were validated by the use of a testbed.

5.2 Summary of Contributions

This thesis has two main contributions:

5.2.1 Self-Configuration of Nodes

Three novel algorithms were proposed and evaluated using both simulation and testbed experimentations. These algorithms were created in order to allow nodes in the network to identify their neighbors, discover their relative location in the network, and select an operating frequency channel.

These algorithms were designed for this specific network where nodes are placed in a static grid topology and where nodes communicate mainly with their neighbors within the same column.

The three algorithms are listed below.

- **Neighbor Identification Algorithm.** Allows each node in the network to find its closest neighbors. Closest neighbors are nodes who can be reached by direct transmission without the need for multi-hop communications. This was done by measuring the RSSI values of the messages sent between the nodes right after they boot. Nodes then use this RSSI value to decide which neighbor(s) are considered their closest neighbors in the network. Which are nodes with the highest RSSI value from the list of neighbors.

This algorithm, is used mainly to reduce any unwanted traffic by limiting the communication to the direct neighbors only. This in turn leads to less interference and less energy consumption, which will lead to the ability of adding more nodes to the network, and increase the scalability of the system.

- **Relative Location Algorithm.** Using this algorithm, nodes will be able to find out what is their location in the network relative to the rest of the nodes. Knowing this relative location will allow each node to decide its role in the network and start its operation based on that role. The role is determined with no interference from the network operator, and by using the same code and the same hardware in every node in the network. This means that the nodes can be completely auto-configured with no prior modification to any specific node in the network.
- **Frequency Allocation Algorithm.** Used to divide the network into small networks of columns operating in a different frequency channel. The allocation is done in order to reduce the interference among the nodes as well as to enable the scalability of the solution. This is also done with no interference from the network operator and no prior planning. Nodes will have to do the planning by themselves.

The evaluation of the algorithms has lead us to conclude that they can be used to allow nodes to configure themselves with no interference from the operator of the network. Results obtained from both the simulation and the testbed experiments have showed that the error in performance decreases as we increase the number of RSSI values used for decision making. Additionally, the number of nodes in the network affects the setup error greatly. However, the value of the error is still acceptable even for high number of simulated columns.

5.2.2 Aggregated Data Collection Technique

The second original contribution of this PhD thesis is a reliable data collection technique that employs data aggregation, polling, and message scheduling to collect the data from nodes within their column. This was shown to lead to the reduction of collisions/retransmissions of messages between the nodes.

Three techniques were proposed and evaluated with and without data aggregation. And from them, one technique was selected that has performed better in the different scenarios that were tested.

The selected polling data collection technique with data aggregation increased the end to end delay, however, as the rate of change in the state of the network is slow, this level of delay can be considered as acceptable. The selected technique was compared with other existing techniques in the literature and was proven, in the specific network topology we are testing, to further reduce the traffic being sent back and forth between the nodes, and increases the reliability of the network.

5.3 Future Research

For future work, a big enough testbed can be deployed in order to test the scalability of the self-configuration algorithms and to be able to validate their performance when comparing with the simulation. This testbed can also be used to test the performance of the Data Collection Technique in a scenario with a higher number of nodes.

Self healing is also a good topic to pursue. For example how does the network fix itself should a setup error happen. Also, adding the ability to detect a node failure during the operation phase.

Another aspect that was not considered in this work is mobility. We have always assumed that the nodes in the network are static. But, with some development, the algorithms can be modified to operate in a mobile network.

Additionally, energy management was not considered throughout this work because we have assumed that the nodes have access to a continuous power source. However, this assumption may not always hold for different types of networks. The energy conservation may be a good topic to address in order to improve the work done in this thesis.

References

[1] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. In *Computer Networks*, volume 52, pages 2292 –2330, 2008.

[2] M. Dohler, D. Barthel, R. Maraninchi, L. Mounier, S. Aubert, C. Dugas, A. Buhrig, R. Paugnat, M. Renaudin, A. Duda, M. Heusse, and R. Valois. The ARESA Project: Facilitating Research, Development and Commercialization of WSNs. In *The 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks. SECON '07*, pages 590 –599, June 2007.

[3] A. Bachir, M. Dohler, T. Watteyne, and K.K. Leung. MAC Essentials for Wireless Sensor Networks. *IEEE Communications Surveys Tutorials*, 12(2):222 –248, 2010.

[4] M. Lehsaini, H. Guyennet, and M. Feham. A novel cluster-based self-organization algorithm for wireless sensor networks. In *International Symposium on Collaborative Technologies and Systems. CTS 2008*, pages 19–26, May 2008.

[5] H. Zhan and J. Yuan. Research and Design of Self-Organized WSN. In *Transaction on Control and Mechanical Systems*, pages 1–5, 2012.

[6] M.M. Abdellatif, J.M. Oliveira, M. Ricardo, and P. Steenkiste. Impact of data collecting techniques on the performance of a Wireless Sensor Network. In *International Symposium on Wireless Communication Systems (ISWCS)*, pages 111–115, Aug 2012.

[7] M.M. Abdellatif, J.M. Oliveira, and M. Ricardo. The effect of data aggregation on the performance of a Wireless Sensor Network employing a polling based data collecting technique. In *IFIP Wireless Days (WD)*, pages 1–6, Nov 2013.

[8] M.M. Abdellatif, J.M. Oliveira, and M. Ricardo. Neighbors and relative location identification using RSSI in a dense wireless sensor network. In *The 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, pages 140–145, June 2014.

[9] R. Braden. Requirements for Internet Hosts - Communication Layers. In *RFC 1122, Internet Engineering Task Force*, October 1989.

[10] R. Braden. Requirements for Internet Hosts - Application and Support. In *RFC 1123, Internet Engineering Task Force*, October 1989.

[11] IEEE Standard for Information Technology- Telecommunications and Information Exchange Between Systems- Local and Metropolitan Area Networks- Specific Requirements Part 3: Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications - Section One. *IEEE Std 802.3-2008 (Revision of IEEE Std 802.3-2005)*, 26:c1–597, 2008.

[12] IEEE Standard for Information Technology– Local and Metropolitan Area Networks– Specific Requirements– Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs). *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003)*, pages 1–320, Sept 2006.

[13] V. G. Cerf and R. E. Icahn. A protocol for packet network intercommunication. *ACM SIGCOMM Computer Communication Review*, 35(2):71–82, 2005.

[14] J. Postel. Internet Protocol. In *RFC 0791, Internet Engineering Task Force*, September 1981.

[15] J. Postel. DoD standard Internet Protocol. In *RFC 0760, Internet Engineering Task Force*, January 1980.

[16] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. In *RFC 2460, Internet Engineering Task Force*, December 1998.

[17] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. In *RFC 1883, Internet Engineering Task Force*, December 1995.

[18] F. Mattern and C. Floerkemeier. From Active Data Management to Event-based Systems and More. pages 242–259. Springer-Verlag, Berlin, Heidelberg, 2010.

[19] S. Thomson, T. Narten, and T. Jinmei. IPv6 Stateless Address Autoconfiguration. In *RFC 4862, Internet Engineering Task Force*, September 2007.

[20] J. Hui and P. Thubert. Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. In *RFC 6282, Internet Engineering Task Force*, September 2011.

[21] C. Bormann. 6LoWPAN Generic Compression of Headers and Headerlike Payloads. In *Internet-Draft draft-bormann-6lowpan-ghc-03, Internet Engineering Task Force*, October 2011. Work in progress.

[22] Z. Shelby, S. Chakrabarti, and E. Nordmark. Neighbor Discovery Optimization for Low Power and Lossy Networks(6LoWPAN). In *Internet-Draft draft-ietf-6lowpan-nd-18, Internet Engineering Task Force*, October 2011. Work in progress.

[23] A. Dunkels, F. Österlind, and Z. He. An Adaptive Communication Architecture for Wireless Sensor Networks. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 335–349. ACM, 2007.

[24] N. Tsiftes, A. Dunkels, Z. He, and T. Voigt. Enabling large-scale storage in sensor networks with the coffee file system. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, pages 349–360. IEEE Computer Society, 2009.

[25] J. Eriksson, A. Dunkels, N. Finne, F. Osterlind, and T. Voigt. Mspsim–an extensible simulator for msp430-equipped sensor boards. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*, page 27, 2007.

[26] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-Level Sensor Network Simulation with COOJA. In *Proceedings of the 31st IEEE Conference on Local Computer Networks*, pages 641 –648, November 2006.

[27] A. Dunkels. Full TCP/IP for 8-bit architectures. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 85–98. ACM, 2003.

[28] A. Dunkels, J. Alonso, and T. Voigt. Making TCP/IP viable for wireless sensor networks. *SICS Research Report*, 2003.

[29] M. Durvy, J. Abeillé, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, and N. Finne. Making sensor networks IPv6 ready. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 421–422. ACM, 2008.

[30] A. Dunkels, O. Schmidt, T. Voigt, and M. Ali. Protothreads: simplifying event-driven programming of memory-constrained embedded systems. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 29–42. ACM, 2006.

[31] Z. Shelby J. Hui and D. Culler. Interoperability Test for 6LoWPAN. *Arch Rock Corporation, Sensinode, Internet Draft (draft-hui-6lowpan-interop-00.txt)*, July 2007.

[32] J. Hui and P. Thubert. Compression format for IPv6 datagrams in 6LoWPAN networks. *draft-ietf-6lowpan-hc-13 (work in progress)*, 2010.

[33] A. Dunkels, B. Gronvall, and T. Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. In *29th Annual IEEE International Conference on Local Computer Networks, 2004.*, pages 455–462. IEEE, 2004.

[34] Tmote Sky. In *http://www.snm.ethz.ch/Projects/TmoteSky*.

[35] K. Mills. A brief survey of self-organization in wireless sensor networks. *Wireless Communications and Mobile Computing*, 7(7):823–834, 2007.

[36] P. Kulkarni, S. Gormus, Z. Fan, and B. Motz. A self-organising mesh networking solution based on enhanced RPL for smart metering communications. In *2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*.

[37] K. Lee and H. Lee. A self-organized and smart-adaptive clustering and routing approach for wireless sensor networks. *International Journal of Distributed Sensor Networks*, 2012.

[38] N. Patwari and A. O. Hero. Using Proximity and Quantized RSS for Sensor Localization in Wireless Networks. In *Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications (WSNA '03)*.

[39] S. J. Halder and W. Kim. A Fusion Approach of RSSI and LQI for Indoor Localization System Using Adaptive Smoothers. *Journal of Computer Networks and Communication, June 2012*.

[40] S. A. Borbash, A. Ephremides, and M. J. McGlynn. An asynchronous neighbor discovery algorithm for wireless sensor networks. *Ad Hoc Networks*, 5(7):998 – 1016, 2007.

[41] F. Wang and J. Liu. Networked Wireless Sensor Data Collection: Issues, Challenges, and Approaches. *IEEE Communications Surveys Tutorials*, 13(4):673 –687, 2011.

[42] V. Pandey, A. Kaur, and N. Chand. A review on data aggregation techniques in wireless sensor network. *Journal of Electronic and Electrical Engineering*, 1(2):01–08, 2010.

[43] H. Zhai and Y. Fang. A distributed packet concatenation scheme for sensor and ad hoc networks. In *MILCOM 2005. IEEE Military Communications Conference.*, pages 1443–1449, 2005.

[44] N. S. Patil and P. R. Patil. Data Aggregation in Wireless Sensor Network. *IEEE International Conference on Computational Intelligence and Computing Research*, 28-29 December 2010.

[45] J. Zheng, X. Xu, and G. Wang. Energy Efficient Data Aggregation Scheduling in Wireless Sensor Networks. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, pages 1662–1667, 2011.

[46] N. Accettura, L.A. Grieco, G. Boggia, and P. Camarda. Performance analysis of the RPL Routing Protocol. In *IEEE International Conference on Mechatronics (ICM)*, pages 767 –772, April 2011.

[47] T. Winter and P. Thubert. RPL: IPv6 Routing Protocol for Low power and Lossy Networks, draft-ietf-roll-rpl-15 (work in progress). In *IETF ROLL working group*, February 2010.

[48] D. Wang, Z. Tao, J. Zhang, and A.A. Abouzeid. RPL Based Routing for Advanced Metering Infrastructure in Smart Grid. In *IEEE International Conference on Communications Workshops (ICC)*, pages 1 –6, May 2010.

[49] SELF-PVP. In *http://www.cmuportugal.org/tiercontent.aspx?id=3374*.

[50] ContikiOS. In *http://www.contiki-os.org/*.

[51] IEEE 802.15.4. In *http://www.ieee802.org/15/pub/TG4.html*.